Logic

Max Schäfer

Formosan Summer School on Logic, Language, and Computation
2010

1 Introduction

This course provides an introduction to the basics of formal logic. We will cover (classical) propositional and first-order logic with their truth-value semantics. We will give an introduction to calculational logic as a tool for reasoning about propositional logic, and to sequent calculus for first-order logic. Finally, we will briefly discuss some limitations of first-order logic.

2 Formal logic

Formal logic as we understand it in these lectures is an approach to making informal mathematical reasoning precise. It has three main ingredients:

- A formal *language* in which to express the mathematical statements we want to reason about.
- A *semantics* that explains the meaning of statements in our formal language in informal terms.
- A *deductive system* that establishes formal rules of reasoning about logical statements which we can apply without having to constantly consider their informal explanation.

It is important to remember that logic (at least as we understand it here) does not allow us to say anything that we would not already be able to express in an informal way. It is simply a way of making informal reasoning precise so as to avoid mistakes and to clarify what assumptions we base our reasoning on.

Different logics may have different languages: for instance, the language of propositional logic, which we will cover first, is a sub-language of the larger language of first-order logic. In both cases the semantics we give for the common parts of the language is the same. There are other variants of propositional and first-order logic (for instance intuitionistic logic) that provide a different semantic explanation for the same formal language.

For a given language and a given semantics, there can still be many different deductive systems that take different approaches to formal reasoning. In these notes, for instance, we will take a look at calculational logic and sequent calculus, both of which are deductive systems for the same kind of logic, but which have a very different look and feel.

3 Classical logic

Since we want to use logic to formalise reasoning about mathematical statements, the concept of *truth* plays a key role: we want our semantics to identify which logical statements are true and which are false, and we want our deductive systems to provide us with rules for deducing true statements.

This approach of viewing logical statements as representing true or false propositions is often called *classical logic*. It accords well with informal mathematical practice and our usual understanding of logic.

Both from philosophical and from practical viewpoints many objections can be raised against the classical interpretation of logic, and many other approaches exist that are more useful than classical logic for certain applications. They will not, however, concern us in these notes.

4 Propositional logic

A very simple logic is *propositional logic*, which formalises reasoning about *atomic propositions*, i.e., statements that are either true or false, although we do not necessarily know which. These propositions are atomic in the sense that we cannot further analyse them. We can, however, connect them in various ways to form other propositions that we also aim to describe in our logic.

4.1 The language of propositional logic

The language of *formulas* of propositional logic is given by the following grammar:

$$\varphi ::= \mathcal{R} \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi$$

The set $\mathcal{R}$ is the set of *propositional letters*, which we use to denote atomic propositions. In these notes, we will use uppercase letters like P, Q and R as propositional letters. The grammar tells us that every propositional letter is already a formula of propositional logic, called an *atomic formula*. Clearly, however, in order to know whether P, viewed as a formula, is true, we need to know whether the atomic proposition it represents is true; this map from propositional letters to atomic propositions forms the basis of the semantics of propositional logic defined in Subsection 4.2 below.

Apart from atomic formulas, every formula contains at least one *connective*, or *logical operator*. We will now give a brief account of their intuitive meaning.

The connective $\bot$ is called *falsity*, and is a propositional formula by itself. Intuitively, it denotes a proposition that is always false, no matter what the propositional letters stand for.

Given two formulas φ and ψ, the formula $\varphi \wedge \psi$ is the *conjunction* of φ and ψ, often read as "φ and ψ". It is only true if both φ and ψ are true, and false otherwise.

Given two formulas φ and ψ, the formula $\varphi \vee \psi$ is the *disjunction* of φ and ψ, often read as "φ or ψ". It is only false if both φ and ψ are false, and true otherwise. Note that this is the so-called "inclusive or" that is true even if both of its alternatives are true.

Given two formulas φ and ψ, the formula $\varphi \to \psi$ is the *implication* of φ and ψ, often read as "φ implies ψ", or (less precisely) "if φ then ψ". It is only false if φ is true and ψ is false, and true otherwise.

The grammar given above is ambiguous. As usual, we use parentheses to disambiguate, and precedence rules to save on parentheses. We will take the order in which the operators were introduced above as giving their precedence, with $\wedge$ binding tightest, and $\to$ least tight. Thus the string $P \wedge Q \to P \vee P \wedge Q$ should be parsed as $((P \wedge Q) \to (P \vee (P \wedge Q)))$, since both $\wedge$ and $\vee$ bind tighter than $\to$, and $\wedge$ tighter than $\vee$.

By convention, all binary connectives associate to the left, except $\to$, which associates to the right. Thus,

$$(P \to Q \to R) \to (P \to Q) \to Q \to R$$

is the same as

$$((P \to (Q \to R)) \to ((P \to Q) \to (Q \to R)))$$

We also define three abbreviations:

1. $\top := \bot \to \bot$
2. $\neg\varphi := \varphi \to \bot$ for any formula φ
3. $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$ for any formulas φ and ψ

We give $\neg$ a higher precedence than $\wedge$, and $\leftrightarrow$ a lower precedence than $\to$. We also agree that $\leftrightarrow$ associates to the left.

We write $\varphi \equiv \psi$ to indicate that formulas φ and ψ are syntactically the same. This disregards parentheses and abbreviations, thus $(P \wedge Q) \vee \neg R \equiv P \wedge Q \vee (R \to \bot)$, but $P \vee Q \wedge R \not\equiv (P \vee Q) \wedge R$.

4.2 Semantics of propositional logic

Now that we have established the language of propositional formulas in which we want to reason, we need to set down its meaning. Since we use formulas to express propositions, and propositions are characterised by whether they are true or false, it will suffice for every formula to define whether it is true or false.

Obviously, we cannot in general say whether a formula is true or false without further information about the propositional letters. This is provided by an interpretation:

Definition 1. *An* interpretation *is a mapping* $I\colon \mathcal{R} \to \mathcal{B}$ *from the set of propositional letters* $\mathcal{R}$ *to the set* $\mathcal{B} := \{\mathtt{F}, \mathtt{T}\}$ *of truth values.*

An interpretation describes a situation in which those propositions that are mapped to `T` are true, whereas those that are mapped to `F` are false.

Definition 2. *Given an interpretation* I*, the semantics* $[\![\varphi]\!]_I \in \mathcal{B}$ *of a propositional formula* φ *is defined by recursion:*

1. *For every propositional letter* $r \in \mathcal{R}$*, we define* $[\![r]\!]_I := I(r)$*.*
2. $[\![\bot]\!]_I := \mathtt{F}$
3. *For two propositional formulas* φ *and* ψ *we define*
   (a) $[\![\varphi \wedge \psi]\!]_I := \mathtt{T}$ *if* $[\![\varphi]\!]_I = \mathtt{T}$ *and* $[\![\psi]\!]_I = \mathtt{T}$*, and* $[\![\varphi \wedge \psi]\!]_I := \mathtt{F}$ *otherwise.*
   (b) $[\![\varphi \vee \psi]\!]_I := \mathtt{F}$ *if* $[\![\varphi]\!]_I = \mathtt{F}$ *and* $[\![\psi]\!]_I = \mathtt{F}$*, and* $[\![\varphi \vee \psi]\!]_I := \mathtt{T}$ *otherwise.*
   (c) $[\![\varphi \to \psi]\!]_I := \mathtt{F}$ *if* $[\![\varphi]\!]_I = \mathtt{T}$ *and* $[\![\psi]\!]_I = \mathtt{F}$*, and* $[\![\varphi \to \psi]\!]_I := \mathtt{T}$ *otherwise.*

These definitions roughly correspond to the intuitive explanations we have given above, except maybe for implication. Although it is common to read $\varphi \to \psi$ as "if φ then ψ", this is not very helpful for understanding its semantics. After all, $\varphi \to \psi$ is just a formula whose truth value is defined in terms of the truth values of φ and ψ; in no sense are we deriving the truth of ψ from the truth of φ. Perhaps the most unexpected feature of implication is that $\varphi \to \psi$ is true if φ is false, no matter if ψ is true or not.

Lemma 1. *For any interpretation* I *we have* $[\![\top]\!]_I = \mathtt{T}$*, and for any formulas* φ *and* ψ *we have*

- $[\![\neg\varphi]\!]_I = \mathtt{T}$ *if* $[\![\varphi]\!]_I = \mathtt{F}$*, and* $[\![\neg\varphi]\!]_I = \mathtt{F}$ *otherwise*
- $[\![\varphi \leftrightarrow \psi]\!]_I = \mathtt{T}$ *if* $[\![\varphi]\!]_I = [\![\psi]\!]_I$*, and* $[\![\varphi \leftrightarrow \psi]\!]_I = \mathtt{F}$ *otherwise.*

Proof. By unfolding the definitions. □

Thus $\top$ can be read as "true", $\neg\varphi$ as "not φ" and $\varphi \leftrightarrow \psi$ as "φ has the same truth value as ψ" or "φ equivales ψ".

Normally, we are not interested in finding out whether a formula is true under a given interpretation: given the above definition, that can be done by a simple calculation. Rather, we are interested in formulas that are true no matter what truth values we assign to them, or how the truth values of two formulas relate independently of any concrete interpretation.

Definition 3. *Let I be an interpretation and φ a propositional formula.*

We say that I is a model *for for φ and write $I \models \varphi$ if $[\![\varphi]\!]_I = \mathtt{T}$.*

We call φ satisfiable *if there is some interpretation I with $I \models \varphi$, and* unsatisfiable *otherwise.*

We call φ valid *or a* tautology *if for all interpretations I we have $I \models \varphi$.*

We say that φ entails *some propositional formula ψ and write $\varphi \Rightarrow \psi$ if for any interpretation I such that $I \models \varphi$ we also have $I \models \psi$. If $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$ we say that φ and φ are* equivalent *and write $\varphi \Leftrightarrow \psi$.*

Example 1. *The formula $P \to P$ is valid. Indeed, let an interpretation I be given. Then either $I(P) = \mathtt{T}$ or $I(P) = \mathtt{F}$. Checking the definition, we can see that in both cases $[\![P \to P]\!]_I = \mathtt{T}$. Since I is arbitrary, this shows that $P \to P$ is valid.*

Example 2. *The formula $P \wedge \neg P$ is unsatisfiable. Indeed, let an interpretation I be given. Then either $I(P) = \mathtt{T}$ or $I(P) = \mathtt{F}$. In the former case, $[\![P]\!]_I = \mathtt{T}$ and $[\![\neg P]\!]_I = \mathtt{F}$, so $[\![P \wedge \neg P]\!]_I = \mathtt{F}$; in the latter case, $[\![P]\!]_I = \mathtt{F}$ and $[\![\neg P]\!]_I = \mathtt{T}$, so again $[\![P \wedge \neg P]\!]_I = \mathtt{F}$.*

Example 3. *The formula $P \wedge Q$ entails the formula $P \vee Q$, i.e., $P \wedge Q \Rightarrow P \vee Q$. Indeed, let an interpretation I be given and assume $I \models P \wedge Q$; then $[\![P]\!]_I = \mathtt{T}$ and $[\![Q]\!]_I = \mathtt{T}$, which means that $I(P) = I(Q) = \mathtt{T}$. The definition then shows that $[\![P \vee Q]\!]_I = \mathtt{T}$, so $I \models P \vee Q$.*

These properties are connected in the following way:

Theorem 2. *φ is valid iff*[1] *$\neg\varphi$ is unsatisfiable iff $\varphi \Leftrightarrow \top$; furthermore, we have $\varphi \Leftrightarrow \psi$ iff $[\![\varphi]\!]_I = [\![\psi]\!]_I$ for every interpretation I.*

Proof. Assume φ is valid, and let any interpretation I be given; then $[\![\varphi]\!]_I = \mathtt{T}$, so $[\![\neg\varphi]\!]_I = \mathtt{F}$. Since I was arbitrary this means that $\neg\varphi$ is unsatisfiable if φ is valid.

Conversely, assume $\neg\varphi$ is unsatisfiable and let an arbitrary interpretation I be given; then $[\![\neg\varphi]\!]_I = \mathtt{F}$, so $[\![\varphi]\!]_I = \mathtt{T}$. Since I was arbitrary this means that φ is valid if $\neg\varphi$ is unsatisfiable.

We can show that φ is valid iff $\varphi \Leftrightarrow \top$ in a very similar manner.

For the second half of the theorem, first assume $\varphi \Leftrightarrow \psi$, and let an interpretation I be given. If $[\![\varphi]\!]_I = \mathtt{T}$, then we must have $[\![\psi]\!]_I = \mathtt{T}$ since $\varphi \Rightarrow \psi$. If $[\![\varphi]\!]_I = \mathtt{F}$, we cannot have $[\![\psi]\!]_I = \mathtt{T}$ as this would violate our assumption that $\psi \Rightarrow \varphi$, so we must have $[\![\psi]\!]_I = \mathtt{F}$. Hence in either case $[\![\varphi]\!]_I = [\![\psi]\!]_I$.

Conversely, assume $[\![\varphi]\!]_I = [\![\psi]\!]_I$ for any interpretation I, and assume $I \models \varphi$; then clearly $I \models \psi$ since $[\![\psi]\!]_I = [\![\varphi]\!]_I = \mathtt{T}$, so $\varphi \Rightarrow \psi$; by the same reasoning we get $\psi \Rightarrow \varphi$, and thus $\varphi \Leftrightarrow \psi$. □

The preceding lemma is what we call a *meta-theorem*: it is a statement *about* the logic we are investigating, which we prove to be true by an informal argument, not by formalising it in the logic itself.

[1] Short for "if and only if".

Example 4. *The formulas $\neg\neg P$ and P are equivalent: by applying Lemma 1 twice, we see that $[\![\neg\neg P]\!]_I = [\![P]\!]_I$ for any interpretation I, hence by Theorem 2 $\neg\neg P \Leftrightarrow P$.*

Note $\varphi \equiv \psi$ and $\varphi \Leftrightarrow \psi$ mean very different things. The former says that φ and ψ are *syntactically* the same formula, if we unfold abbreviations and fully parenthesise them. The latter says that they are *semantically* the same in the sense that they have the same truth value in every interpretation. If $\varphi \equiv \psi$ then also $\varphi \Leftrightarrow \psi$, but not necessarily the other way around; for instance, $\neg\neg P \not\equiv P$.

4.3 Truth tables

We have defined the meaning of complex formulas in terms of the meanings of their components, and the meaning of the most basic formulas in terms of the truth values assigned to propositional letters by the interpretation. For this reason, it seems clear that the truth value we compute for a formula only depends on the truth values that the interpretation assigns to those propositional letters that actually occur in the formula.

For instance, the truth value of $P \vee \neg Q$ certainly only depends on $I(P)$ and $I(Q)$. It is the same no matter whether $I(R) = \mathtt{T}$ or $I(R) = \mathtt{F}$. In order to make this intuition precise, we first need to define what it means for a propositional letter to "occur" in a formula.

Definition 4. *The set $\mathtt{PL}(\varphi)$ of propositional letters occuring in a formula φ is defined as follows:*

1. *For every propositional letter $r \in \mathcal{R}$, we define $\mathtt{PL}(r) := \{r\}$.*
2. *$\mathtt{PL}(\bot) := \emptyset$.*
3. *For two propositional formulas φ and ψ we define*
   (a) *$\mathtt{PL}(\varphi \wedge \psi) := \mathtt{PL}(\varphi) \cup \mathtt{PL}(\psi)$*
   (b) *$\mathtt{PL}(\varphi \vee \psi) := \mathtt{PL}(\varphi) \cup \mathtt{PL}(\psi)$*
   (c) *$\mathtt{PL}(\varphi \to \psi) := \mathtt{PL}(\varphi) \cup \mathtt{PL}(\psi)$.*

Definition 5. *Two interpretations I_1 and I_2 are said to* agree *on a set $R \subseteq \mathcal{R}$ of propositional letters if $I_1(r) = I_2(r)$ for every $r \in R$.*

Lemma 3 (Agreement)**.** *For any formula φ and interpretations I_1 and I_2 that agree on $\mathtt{PL}(\varphi)$ we have $[\![\varphi]\!]_{I_1} = [\![\varphi]\!]_{I_2}$.*

Proof. Let an arbitrary formula φ and two interpretations I_1 and I_2 be given that agree on $\mathtt{PL}(\varphi)$. We prove $[\![\varphi]\!]_{I_1} = [\![\varphi]\!]_{I_2}$ by induction on the structure of φ.

If φ is a propositional letter r, then $[\![\varphi]\!]_{I_1} = I_1(r)$ and $[\![\varphi]\!]_{I_2} = I_2(r)$. Since I_1 and I_2 agree on $\mathtt{PL}(\varphi) = \{r\}$ we must have $I_1(r) = I_2(r)$, and hence $[\![\varphi]\!]_{I_1} = [\![\varphi]\!]_{I_2}$.

If φ is $\bot$, then $[\![\varphi]\!]_{I_1} = \mathtt{F} = [\![\varphi]\!]_{I_2}$.

If φ is of the form $\psi \wedge \chi$, then $\mathtt{PL}(\varphi) = \mathtt{PL}(\psi) \cup \mathtt{PL}(\chi)$. Since I_1 and I_2 agree on $\mathtt{PL}(\varphi)$, they also agree on $\mathtt{PL}(\psi)$ and $\mathtt{PL}(\chi)$, so by induction hypothesis we have $[\![\psi]\!]_{I_1} = [\![\psi]\!]_{I_2}$ and $[\![\chi]\!]_{I_1} = [\![\chi]\!]_{I_2}$. To show $[\![\varphi]\!]_{I_1} = [\![\varphi]\!]_{I_2}$, assume $[\![\varphi]\!]_{I_1} = \mathtt{T}$; then $[\![\psi]\!]_{I_1} = \mathtt{T}$ and $[\![\chi]\!]_{I_1} = \mathtt{T}$, so $[\![\psi]\!]_{I_2} = \mathtt{T}$ and $[\![\chi]\!]_{I_2} = \mathtt{T}$, and hence $[\![\varphi]\!]_{I_2} = \mathtt{T}$. Conversely, assume $[\![\varphi]\!]_{I_2} = \mathtt{T}$, then $[\![\psi]\!]_{I_2} = \mathtt{T}$ and $[\![\chi]\!]_{I_2} = \mathtt{T}$, so $[\![\psi]\!]_{I_1} = \mathtt{T}$ and $[\![\chi]\!]_{I_1} = \mathtt{T}$, and hence $[\![\varphi]\!]_{I_1} = \mathtt{T}$.

If φ is of the form $\psi \vee \chi$ or $\psi \to \chi$, the argument is similar.

It follows by induction that $[\![\varphi]\!]_{I_1} = [\![\varphi]\!]_{I_2}$. □

While the agreement lemma may seem like a rather technical result, it is actually quite important in practice, since it shows that whenever we want to find out whether a formula is valid or satisfiable, it suffices to consider the possible interpretations of the propositional letters that occur in the formula only.

This makes a big difference: while in general there can be infinitely many propositional letters, and hence infinitely many interpretations, if our formula only makes use of, say, n propositional letters, we only have to consider the different ways that truth values could be assigned to those n letters, of which there are 2^n.

We can represent these interpretations by *truth tables* like the following:

P	Q	$P \to Q$	$(P \to Q) \to P$	$((P \to Q) \to P) \to P$
F	F	T	F	T
F	T	T	F	T
T	F	F	T	T
T	T	T	T	T

Here, the first two columns give the truth values assigned to P and Q in different interpretations, hence every line represents one of the $2^2 = 4$ interpretations we need to consider. In every further column, we compute the truth values of more complex formulas under each of these interpretations, building up from simple formulas to more complex ones.

Since the last column only contains T, we know that *Peirce's Law*, the formula $((P \to Q) \to P) \to P$, is valid. We can also use truth tables to establish the equivalence of two formulas:

P	Q	$\neg P$	$\neg Q$	$P \wedge Q$	$\neg(P \wedge Q)$	$\neg P \vee \neg Q$
F	F	T	T	F	T	T
F	T	T	F	F	T	T
T	F	F	T	F	T	T
T	T	F	F	T	F	F

Since the last two columns show the same truth value in every interpretation, we see that $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$, which is one of *De Morgan's Laws*, the other being $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$.

Theorem 4. *For a propositional formula φ, it is decidable whether φ is satisfiable, and also whether it is valid.*

Proof. Given φ, it suffices to draw up a truth table. If the last column contains a T, then φ is satisfiable; if it contains only T, then φ is valid. □

Notice that for a formula containing n propositional letters we need to check 2^n possible assignments, so this algorithm for deciding satisfiability is exponential in n. In fact, the problem of deciding satisfiability is the paradigmatic example of an NP-complete problem; whether it is possible to solve NP-complete problems in polynomial time is the most famous open problem of theoretical computer science. There are heuristic algorithms that make it possible to decide the satisfiability of *most* propositional formulas very efficiently, although their worst-case complexity is still exponential.

4.4 Logical equivalence

We have above defined the concept of logical equivalence by saying that $\varphi \Leftrightarrow \psi$ if $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$, i.e., every model of φ is also a model of ψ and vice versa.

Using the definition, it is easy to check that equivalence is reflexive, symmetric and transitive, i.e., for all formulas φ, ψ, χ we have $\varphi \Leftrightarrow \varphi$; if $\varphi \Leftrightarrow \psi$ then $\psi \Leftrightarrow \varphi$; if $\varphi \Leftrightarrow \psi$ and $\psi \Leftrightarrow \chi$ then $\varphi \Leftrightarrow \chi$.

The following result shows an interesting connection between $\leftrightarrow$ and $\Leftrightarrow$, which allows us to reflect the meta-theoretic $\Leftrightarrow$ operator back into the logic:

Lemma 5. $\models \varphi \leftrightarrow \psi$ *iff* $\varphi \Leftrightarrow \psi$

Proof. Follows from Lemma 1 and Theorem 2. □

Equivalent formulas have the same truth value under every interpretation, so we can substitute one for the other without changing the meaning of a formula.

To carefully prove this statement, we need to first define what it means to substitute one formula for another. To keep things simple, we only define what it means to subsitute a formula for a propositional letter.

Definition 6. *The substitution $\varphi[\vartheta/r]$ of a formula ϑ for all occurrences of a propositional letter r in formula φ is defined by structural recursion as follows:*

1. *if φ is some $r' \in \mathcal{R}$, then*
   (a) $\varphi[\vartheta/r] := \vartheta$ *if* $r = r'$
   (b) $\varphi[\vartheta/r] := r'$ *otherwise*
2. *if φ is $\bot$, then* $\varphi[\vartheta/r] := \bot$
3. *if φ is $\psi \wedge \chi$, then* $\varphi[\vartheta/r] := \psi[\vartheta/r] \wedge \chi[\vartheta/r]$
4. *if φ is $\psi \vee \chi$, then* $\varphi[\vartheta/r] := \psi[\vartheta/r] \vee \chi[\vartheta/r]$
5. *if φ is $\psi \to \chi$, then* $\varphi[\vartheta/r] := \psi[\vartheta/r] \to \chi[\vartheta/r]$.

As a simple example, $(R \to R)[Q \wedge \neg Q/R]$ is $(Q \wedge \neg Q) \to (Q \wedge \neg Q)$.
Now we can prove the following meta-theorem:

Theorem 6 (Leibniz' Law for Propositions)**.** *If $\psi_1 \Leftrightarrow \psi_2$, then $\varphi[\psi_1/r] \Leftrightarrow \varphi[\psi_2/r]$ for any propositional letter r and formulas φ, ψ_1, ψ_2.*

Proof. Exercise (by structural induction on φ). □

Another important result is the following, ostensibly very similar theorem:

Theorem 7 (Substitution in Tautologies)**.** *If $\models \varphi$, then $\models \varphi[\psi/r]$ for any propositional letter r and formulas φ, ψ.*

Proof. Let an interpretation I be given. We define the interpretation I' to be the same as I, except that it assigns the value $[\![\psi]\!]_I$ to r. It can now be shown by structural induction on φ that $[\![\varphi[\psi/r]]\!]_I = [\![\varphi]\!]_{I'}$ (exercise!).

Now let an arbitrary interpretation I be given. Since φ is valid, we have $[\![\varphi]\!]_{I'} = \mathtt{T}$, so $[\![\varphi[\psi/r]]\!]_I = \mathtt{T}$. But I was arbitrary, so $\models \varphi[\psi/r]$. □

4.5 Functionally complete sets of connectives

Using truth tables and Theorem 7, it is easy to establish a long list of equivalences between different operators, for instance:

- $\bot \Leftrightarrow \varphi \wedge \neg\varphi$
- $\neg\neg\varphi \Leftrightarrow \varphi$
- $\varphi \wedge \psi \Leftrightarrow \neg(\neg\varphi \vee \neg\psi)$
- $\varphi \vee \psi \Leftrightarrow \neg(\neg\varphi \wedge \neg\psi)$
- $\varphi \vee \psi \Leftrightarrow \neg\varphi \to \psi$

These equivalences show that some operators can be defined in terms of others. For example, by the substitution theorem and the last equivalence above, wherever we write $\varphi \vee \psi$ we could just as well write $\neg\varphi \to \psi$. In particular, the above equivalences show that every operator can be expressed in terms of $\to$ and $\bot$.

In a sense, then, our set of operators is too large in that we can throw out certain operators without diminishing the set of propositions we can describe. Conversely, one might ask whether the set of operators is large enough, or whether we are missing any.

Since ultimately our semantics is defined in terms of truth values of atomic propositions, we can make this question precise in the following way:

Definition 7. *Let us call a function $f : \mathcal{B}^n \to \mathcal{B}$ that maps any n-tuple of truth values to a truth value a* truth function *of arity n.*

Every formula φ with $\mathtt{PL}(\varphi) = \{r_1, \ldots, r_n\}$ expresses a truth function f_φ of arity n defined as follows:

$$f_\varphi(x_1, \ldots, x_n) := [\![\varphi]\!]_{I_{r_1:=x_1,\ldots,r_n:=x_n}}$$

where $I_{r_1:=x_1,\ldots,r_n:=x_n}$ is the interpretation that assigns x_1 to r_1, x_2 to r_2, and so on, and assigns arbitrary values to all other propositional letters.

For instance, the formula $P \wedge Q$ expresses the binary truth function that maps $(\mathtt{T}, \mathtt{T})$ to $\mathtt{T}$, and any other tuple to $\mathtt{F}$.

The formula $\bot$ expresses the truth function of arity 0 that maps the tuple $()$ to $\mathtt{F}$.

Definition 8. *We call a set of operators* functionally complete for arity n *if every truth function of arity n can be expressed by a formula that is built up only from propositional letters and operators from this set. If a set of operators is functionally complete for any arity, we simply call it functionally complete.*

Lemma 8. *The set $\{\bot, \to, \vee, \wedge\}$ is functionally complete.*

Proof. Let f be a truth function of arity n, and choose n distinct propositional letters $\{r_1, \ldots, r_n\}$.

We prove the result by induction on n.

If $n = 0$, then f is either constant $\mathtt{T}$ or constant $\mathtt{F}$; in the former case, it is expressed by $\top$, which is just $\bot \to \bot$, in the latter case by $\bot$.

If $n > 0$, we define two $(n-1)$-ary functions f^+ and f^- as follows:

$$\begin{aligned} f^+(x_1, \ldots, x_{n-1}) &:= f(x_1, \ldots, x_{n-1}, \mathtt{T}) \\ f^-(x_1, \ldots, x_{n-1}) &:= f(x_1, \ldots, x_{n-1}, \mathtt{F}) \end{aligned}$$

By induction hypothesis we can find formulas φ^+ and φ^- to express f^+ and f^-; we claim that $\varphi := r_n \wedge \varphi^+ \vee \neg r_n \wedge \varphi^-$ expresses f. Indeed, let truth values $t_1, \ldots, t_n$ be given and construct $I := I_{r_1:=t_1,\ldots,r_n:=t_n}$ as above. We claim that $[\![\varphi]\!]_I = f(t_1, \ldots, t_n)$.

If $t_n = \mathtt{T}$, then $f(t_1, \ldots, t_n) = f^+(t_1, \ldots, t_n)$, and also $[\![\varphi]\!]_I = [\![\varphi^+]\!]_I$, but the two right-hand sides are equal by induction hypothesis. If $t_n = \mathtt{F}$, then $f(t_1, \ldots, t_n) = f^-(t_1, \ldots, t_n) = [\![\varphi^-]\!]_I = [\![\varphi]\!]_I$. □

There is another way to prove this result, which can easily be visualised using truth tables. Assume we want to find a formula to express the function f encoded by the following table:

P	Q	R	f
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

We see that it is true in the fourth, sixth, seventh and eighth lines. Now we simply take these lines, and for every line build a conjunction of P, Q and R, where each propositional letter occurs negated if its value in that line is `F`, and un-negated otherwise. For the fourth line, for example, we get $\neg P \wedge Q \wedge R$. Then we take the disjunction of all these lines to arrive at the formula

$$\neg P \wedge Q \wedge R \vee P \wedge \neg Q \wedge R \vee P \wedge Q \wedge \neg R \vee P \wedge Q \wedge R$$

which expresses the given truth function.

Once we know that $\{\bot, \rightarrow, \vee, \wedge\}$ is functionally complete, we can easily show the same for other sets; for instance, $\{\bot, \rightarrow, \vee\}$ by itself is already functionally complete, and so is $\{\bot, \rightarrow\}$. On the other hand, $\{\bot, \vee\}$ is not functionally complete: for instance, it is not hard to show that any formula φ with a single propositional letter r built up from $\bot$ and $\vee$ is equivalent to either r or $\bot$ or $\top$, so there is no way to express negation.

5 Calculational logic

Drawing up truth tables is a tedious and error-prone business. The approach of calculational logic tries to avoid reasoning by cases as much as possible, instead deriving the truth of logical formulas from the truth of other, simpler formulas by "equational" reasoning.

Calculational derivations are based on a set of *laws*, which are just equivalence formulas that we know to be valid (for instance, we can check them using truth tables). The precise set of laws to be used is not so important, but the main appeal of calculational logic comes from keeping this set as small as possible, and deriving other laws from it. In this section, we will use the set of laws from Roland Backhouse's book "Program Construction", shown in Figure 1.

All of these laws, except the Substitution law, are of the form $\varphi \leftrightarrow \psi$, and in every case we can use truth tables to check that the given formula is valid.

By Lemma 5 these laws can also be read as semantic equivalences of the form $\varphi \Leftrightarrow \psi$; by Theorem 7, they hold with arbitrary formulas substituted for the propositional letters; finally, by Theorem 6 we can substitute their right hand side for their left hand side in any other formula without changing its validity.

Here is an example of a derivation using the laws of calculational logic:

$$\begin{array}{ll} & \neg(P \leftrightarrow Q) \\ \Leftrightarrow & \quad \{ \text{ unfolding } \neg \ \} \\ & (P \leftrightarrow Q) \leftrightarrow \bot \\ \Leftrightarrow & \quad \{ \text{ associativity of } \leftrightarrow \ \} \\ & P \leftrightarrow (Q \leftrightarrow \bot) \\ \Leftrightarrow & \quad \{ \text{ unfolding } \neg \ \} \\ & P \leftrightarrow \neg Q \end{array}$$

Associativity of $\leftrightarrow$	$((P \leftrightarrow Q) \leftrightarrow R) \leftrightarrow (P \leftrightarrow (Q \leftrightarrow R))$
Symmetry of $\leftrightarrow$	$P \leftrightarrow Q \leftrightarrow Q \leftrightarrow P$
Unfolding $\top$	$\top \leftrightarrow P \leftrightarrow P$
Unfolding $\neg$	$\neg P \leftrightarrow P \leftrightarrow \bot$
Idempotence of $\vee$	$P \vee P \leftrightarrow P$
Symmetry of $\vee$	$P \vee Q \leftrightarrow Q \vee P$
Associativity of $\vee$	$P \vee (Q \vee R) \leftrightarrow (P \vee Q) \vee R$
Distributivity of $\vee$	$P \vee (Q \leftrightarrow R) \leftrightarrow P \vee Q \leftrightarrow P \vee R$
Excluded Middle	$P \vee \neg P \leftrightarrow \top$
Golden Rule	$P \wedge Q \leftrightarrow P \leftrightarrow Q \leftrightarrow P \vee Q$
Unfolding $\rightarrow$	$P \rightarrow Q \leftrightarrow Q \leftrightarrow P \vee Q$
Substitution	$(P \leftrightarrow Q) \wedge \varphi[P/R] \leftrightarrow (P \leftrightarrow Q) \wedge \varphi[Q/R]$

Figure 1: Laws of calculational logic

This derivation can be read as a series of steps, where each step shows the semantic equivalence of two formulas with a comment between curly braces mentioning the applied law. We will often omit steps applying associativity like the second step above, and instead write associative operators without parentheses.

By transitivity of $\Leftrightarrow$, the derivation shows that $\neg(P \leftrightarrow Q) \Leftrightarrow P \leftrightarrow \neg Q$.

Here is another derivation involving negation:

$$
\begin{array}{ll}
 & \neg\neg P \\
\Leftrightarrow & \quad \{ \text{ unfolding } \neg \text{ twice } \} \\
 & P \leftrightarrow \bot \leftrightarrow \bot \\
\Leftrightarrow & \quad \{ \text{ unfolding } \top \} \\
 & P \leftrightarrow \top \\
\Leftrightarrow & \quad \{ \text{ unfolding } \top \} \\
 & P
\end{array}
$$

Perhaps the most complex of the laws of calculational logic is the Golden Rule. We can use it to derive the important *Absorption Law*:

$$
\begin{array}{ll}
 & P \vee (P \wedge Q) \\
\Leftrightarrow & \quad \{ \text{ Golden Rule } \}
\end{array}
$$

$$
\begin{array}{ll}
 & P \vee (P \leftrightarrow Q \leftrightarrow P \vee Q) \\
\Leftrightarrow & \quad \{ \text{ distributivity of } \vee \text{ twice } \} \\
 & P \vee P \leftrightarrow P \vee Q \leftrightarrow P \vee P \vee Q \\
\Leftrightarrow & \quad \{ \text{ idempotence of } \vee \text{ twice } \} \\
 & P \leftrightarrow P \vee Q \leftrightarrow P \vee Q \\
\Leftrightarrow & \quad \{ \text{ unfolding } \top \} \\
 & P \leftrightarrow \top \\
\Leftrightarrow & \quad \{ \text{ unfolding } \top \} \\
 & P
\end{array}
$$

We usually deal with implication by the unfolding rule:

$$
\begin{array}{ll}
 & \top \rightarrow P \\
\Leftrightarrow & \quad \{ \text{ unfolding } \rightarrow \} \\
 & P \leftrightarrow P \vee \top \\
\Leftrightarrow & \quad \{ P \vee \top \leftrightarrow \top \text{ (exercise!) } \} \\
 & P \leftrightarrow \top \\
\Leftrightarrow & \quad \{ \text{ unfolding } \top \} \\
 & P
\end{array}
$$

The substitution rule is not a single law, but a whole set of laws, one for every formula φ. We can show by structural induction on φ that the rule is valid for every choice of φ. Here is an example for the use of this rule, where we make use of the equivalence just established:

$$
\begin{array}{ll}
 & P \wedge (P \rightarrow Q) \\
\Leftrightarrow & \quad \{ \text{ unfolding } \top \} \\
 & (P \leftrightarrow \top) \wedge (P \rightarrow Q) \\
\Leftrightarrow & \quad \{ \text{ substitution } \} \\
 & (P \leftrightarrow \top) \wedge (\top \rightarrow Q) \\
\Leftrightarrow & \quad \{ \top \rightarrow Q \leftrightarrow Q \text{ and unfolding } \top \} \\
 & P \wedge Q
\end{array}
$$

6 First-order logic

While propositional logic has its uses in circuit design and verification, it is by itself not sufficient to formalise many mathematical statements. In mathematics we often want to express propositions about individuals, for instance the statement "for every x we have $1 + x > x$".

In the example statement, the individuals are numbers, ranged over by *individual variables* such as x. The symbol 1, on the other hand, is not a variable: it is a constant. Both constants and individual variables are terms; more complex terms can be built up by using functions, such as the binary function $+$. Relations such as the binary $>$ are then used to express atomic propositions about terms, which we can combine into more complex propositions. First-order logic formalises statements like this in an abstract setting.

6.1 The language of first-order logic

Definition 9. *A first-order* signature $\Sigma = \langle \mathcal{F}, \mathcal{R} \rangle$ *consists of*

- function symbols $f \in \mathcal{F}$ *with arity* $\alpha(f) \in \mathbb{N}$
- relation symbols $r \in \mathcal{R}$ *with arity* $\alpha(r) \in \mathbb{N}$

We write f/n *to mean* $\alpha(f) = n$*, and* $\mathcal{F}^{(n)} := \{f/n \mid f \in \mathcal{F}\}$*, same for* $\mathcal{R}^{(n)}$*.*

Definition 10. *The set of terms* $\mathcal{T}(\Sigma, \mathcal{V})$ *over a signature* Σ *and a set* $\mathcal{V}$ *of* individual variables *is inductively defined:*

- $\mathcal{V} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$
- *for* $f/n \in \mathcal{F}$*,* $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$*, also* $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$

For a 0*-ary constant* d*, we write* $d()$ *simply as* d*.*

Example 5. *Consider the signature* $\Sigma_A = \langle \mathcal{F}_A, \mathcal{R}_A \rangle$ *with* $\mathcal{F}_A = \{\boldsymbol{0}/0, s/1, +/2, \times/2\}$ *and* $\mathcal{R}_A = \{\leq\}$*.*

Examples for terms over Σ_A *and* $\mathcal{V} := \{x, y\}$ *are* $\boldsymbol{0}$*,* $s(\boldsymbol{0})$*,* $s(s(\boldsymbol{0}))$*,* $s(x)$*,* $+(s(x), y)$*,* $s(+(x, y))$*. Non-examples are* $\boldsymbol{0}(\boldsymbol{0})$ *and* $+(s(\boldsymbol{0}))$*. We usually write* $+(x, y)$ *as* $x + y$*, and similar for other binary operators, but this is purely syntactic sugar.*

Definition 11. *An* atomic formula*, or* atom *for short, is of the form* $r(t_1, \ldots, t_n)$*, where* $r/n \in \mathcal{R}$*,* $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$*; like before we write just* r *if* $\alpha(r) = 0$*. Additionally, for any two terms* $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$*, there is an atom* $s \doteq t$*. The set of atoms over* Σ *and* $\mathcal{V}$ *is written* $\mathcal{A}(\Sigma, \mathcal{V})$*.*

The language of first-order formulas over Σ *and* $\mathcal{V}$ *is described by the following grammar:*

$$\varphi \quad ::= \quad \mathcal{A}(\Sigma, \mathcal{V}) \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \forall \mathcal{V}.\varphi \mid \exists \mathcal{V}.\varphi$$

We define $\top$, $\neg\varphi$ and $\varphi \leftrightarrow \psi$ as abbreviations in the same way as for propositional logic.

The quantifiers $\forall$ and $\exists$ bind less tight than any of the other connectives.

Note that every propositional formula over the set $\mathcal{R}$ of propositional letters is a first-order formula over $\Sigma = \langle \emptyset, \mathcal{R} \rangle$ and $\mathcal{V} = \emptyset$.

Example 6. *Taking the signature Σ_A and $\mathcal{V}$ from before, the following are atoms (again, we use infix notation):*

- $x \doteq y$
- $x \leq s(x)$
- $x \leq x + y$

And here are some formulas:

- $\neg(x \doteq s(x))$
- $(x \leq s(y)) \rightarrow (x \doteq s(y) \vee x \leq y)$
- $\forall n. \forall m. m \leq m + n$

An occurrence of an individual variable is called *bound* if it is within the scope of a quantifier, otherwise it is *free*. For instance, in the following formula the underlined occurrences are bound, and the others are free:

$$(\forall x. R(\underline{x}, z) \rightarrow (\exists y. S(\underline{y}, \underline{x}))) \wedge T(x)$$

Note that one and the same variable can have both bound and free occurrences in the same formula: in this case, x has one free and one bound occurrence, whereas the only occurrence of y is bound.

A formula without bound variable occurrences is called *closed*. We formally define the set of variables that occur freely (or *free variables* for short) in a formula as follows:

Definition 12. *The set $\mathrm{FV}(t)$ of free variables for a term t is defined by recursion:*

1. $\mathrm{FV}(x) = \{x\}$ *for* $x \in \mathcal{V}$
2. $\mathrm{FV}(f(t_1, \ldots, t_n)) = \bigcup_{i \in \{1,\ldots,n\}} \mathrm{FV}(t_i)$

Likewise, the set $\mathrm{FV}(\varphi)$ of free variables for a formula φ is defined by recursion:

1. $\mathrm{FV}(r(t_1, \ldots, t_n)) = \bigcup_{i \in \{1,\ldots,n\}} \mathrm{FV}(t_i)$
2. $\mathrm{FV}(s \doteq t) = \mathrm{FV}(s) \cup \mathrm{FV}(t)$
3. $\mathrm{FV}(\bot) = \emptyset$

4. $\mathrm{FV}(\varphi \wedge \psi) = \mathrm{FV}(\varphi \vee \psi) = \mathrm{FV}(\varphi \to \psi) = \mathrm{FV}(\varphi) \cup \mathrm{FV}(\psi)$

5. $\mathrm{FV}(\forall x.\varphi) = \mathrm{FV}(\varphi) \setminus \{x\}$

6. $\mathrm{FV}(\exists x.\varphi) = \mathrm{FV}(\varphi) \setminus \{x\}$

Example 7. $\mathrm{FV}(s(x) \doteq 0 \vee x \doteq x) = \{x\}$, $\mathrm{FV}(\forall m.\exists n.s(m) \leq n) = \emptyset$, $\mathrm{FV}(\exists x.s(x) \leq y) = \{y\}$

We can now define substitution of a term for an individual variable:

Definition 13. *The operation of substituting a term t for a variable x in a term s (written $s[t/x]$) is defined as follows:*

1. $y[t/x] = \begin{cases} t & \text{if } x = y, \\ y & \text{otherwise} \end{cases}$

2. $f(t_1, \ldots, t_n)[t/x] = f(t_1[t/x], \ldots, t_n[t/x])$

On formulas, the definition is

1. $r(t_1, \ldots, t_n)[t/x] = r(t_1[t/x], \ldots, t_n[t/x])$

2. $(s_1 \doteq s_2)[t/x] = (s_1[t/x] \doteq s_2[t/x])$

3. $\bot[t/x] = \bot$

4. $(\varphi \circ \psi)[t/x] = (\varphi[t/x] \circ \psi[t/x])$, *where* $\circ$ *is either* $\wedge$ *or* $\vee$ *or* $\to$

5. $(\forall y.\varphi)[t/x] = \begin{cases} \forall y.\varphi & \text{if } x = y, \\ \forall y.(\varphi[t/x]) & \text{if } x \neq y,\ y \notin \mathrm{FV}(t) \end{cases}$

   $(\exists y.\varphi)[t/x] = \begin{cases} \exists y.\varphi & \text{if } x = y, \\ \exists y.(\varphi[t/x]) & \text{if } x \neq y,\ y \notin \mathrm{FV}(t) \end{cases}$

Note that substitution on formulas is a partial operation. We use the same notation as for substitution of formulas for propositional letters in propositional logic, but these two are actually different operations.

Example 8.

- $(s(x) \doteq 0 \vee x \doteq x)[s(0)/x] = (s(s(0)) \doteq 0 \vee s(0) \doteq s(0))$
- $(s(x) \doteq 0 \vee x \doteq x)[s(0)/y] = (s(x) \doteq 0 \vee x \doteq x)$
- $(\forall m.\exists n.m \leq n)[0/m] = (\forall m.\exists n.m \leq n)$
- $(\exists x.x \leq y)[x/y]$ *is not defined*
- $(\forall m.\exists n.m \leq n)[m/n]$ *is not defined*

Definition 14. *We say that the formula* $\forall x.\varphi$ alpha-reduces *to formula* $\forall y.\varphi'$ *if* $\varphi' = \varphi[y/x]$*, and similar for* $\exists x.\varphi$*.*

φ *is called* alpha equivalent *to* ψ *(written* $\varphi \equiv_\alpha \psi$*), if* ψ *results from* φ *by any number of alpha reductions anywhere inside* φ*.*

Example 9.

- $(\forall x.R(x,x)) \equiv_\alpha (\forall y.R(y,y))$
- $(\forall x.\exists x.S(x)) \equiv_\alpha (\forall y.\exists x.S(x)) \equiv_\alpha (\forall y.\exists z.S(z))$
- $(\forall x.\exists y.T(x,y)) \not\equiv_\alpha (\forall x.\exists x.T(x,x))$

6.2 The semantics of first-order logic

Definition 15. *A first-order structure* $\mathcal{M} = \langle D, I\rangle$ *over signature* Σ *comprises*

- *a non-empty set* D*, the* domain
- *an interpretation* $I = \langle [\![\cdot]\!]_\mathcal{F}, [\![\cdot]\!]_\mathcal{R}\rangle$ *such that*
  - *for every* $f \in \mathcal{F}^{(n)}$*,* $[\![f]\!]_\mathcal{F} \colon D^n \to D$
  - *for every* $r \in \mathcal{R}^{(n)}$*,* $[\![r]\!]_\mathcal{R} \colon D^n \to \mathbb{B}$

A variable assignment *on* I *is a function* $\sigma \colon \mathcal{V} \to D$*.*

Definition 16. *We write* $\sigma[x := t]$ *for the assignment* σ' *such that* $\sigma'(x) = t$*, and* $\sigma'(y) = \sigma(y)$ *for all* $y \neq x$*.*

Definition 17. *The semantics* $[\![t]\!]_{\mathcal{M},\sigma} \in D$ *of a term* t *over a structure* $\mathcal{M}$ *and a variable assignment* σ *is defined as follows:*

1. $[\![x]\!]_{\mathcal{M},\sigma} = \sigma(x)$
2. $[\![f(t_1,\ldots,t_n)]\!]_{\mathcal{M},\sigma} = [\![f]\!]_\mathcal{F}([\![t_1]\!]_{\mathcal{M},\sigma},\ldots,[\![t_n]\!]_{\mathcal{M},\sigma})$

For formulas, we define

1. $[\![r(t_1,\ldots,t_n)]\!]_{\mathcal{M},\sigma} = [\![r]\!]_\mathcal{R}([\![t_1]\!]_{\mathcal{M},\sigma},\ldots,[\![t_n]\!]_{\mathcal{M},\sigma})$
2. $[\![s \doteq t]\!]_{\mathcal{M},\sigma} = \mathtt{T}$ *if* $[\![s]\!]_{\mathcal{M},\sigma} = [\![t]\!]_{\mathcal{M},\sigma}$*, otherwise* $[\![s \doteq t]\!]_{\mathcal{M},\sigma} = \mathtt{F}$
3. $[\![\bot]\!]_{\mathcal{M},\sigma} = \mathtt{F}$
4. $[\![\varphi \wedge \psi]\!]_{\mathcal{M},\sigma}$*, etc.: as before*
5. $[\![\forall x.\varphi]\!]_{\mathcal{M},\sigma} = \begin{cases} \mathtt{T} & \text{if, for all } d \in D,\ [\![\varphi]\!]_{\mathcal{M},\sigma[x:=d]} = \mathtt{T}, \\ \mathtt{F} & \text{otherwise} \end{cases}$
6. $[\![\exists x.\varphi]\!]_{\mathcal{M},\sigma} = \begin{cases} \mathtt{T} & \text{if there is } d \in D \text{ with } [\![\varphi]\!]_{\mathcal{M},\sigma[x:=d]} = \mathtt{T}, \\ \mathtt{F} & \text{otherwise} \end{cases}$

Definition 18. *We write* $\mathcal{M}, \sigma \models \varphi$ *for* $[\![\varphi]\!]_{\mathcal{M},\sigma} = \mathtt{T}$. $\mathcal{M} \models \varphi$ *means that* $\mathcal{M}, \sigma \models \varphi$ *for any* σ*;* $\mathcal{M}$ *is then called a* model *for* φ*. If* φ *has a model, we call it* satisfiable*. If every* $\mathcal{M}$ *is a model for* φ*, we call it* valid *and write* $\models \varphi$*.*

For a set of formulas Γ *we extend these definitions point-wise: We write* $\mathcal{M}, \sigma \models \Gamma$ *to mean that* $\mathcal{M}, \sigma \models \gamma$ *for every* $\gamma \in \Gamma$*, and similar for* $\mathcal{M} \models \Gamma$*. We call a set of formulas satisfiable if it has a model and valid if every structure is a model for it.*

Finally, we write $\Gamma \models \varphi$ *to mean that any* $\mathcal{M}$ *and* σ *such that* $\mathcal{M}, \sigma \models \Gamma$ *also gives* $\mathcal{M}, \sigma \models \varphi$*.*

Example 10. *Consider the structure* $\mathcal{M} = \langle \mathbb{N}, \langle [\![\cdot]\!]_{\mathcal{F}}, [\![\cdot]\!]_{\mathcal{R}} \rangle \rangle$ *for signature* Σ_A *as before:*

- $[\![\boldsymbol{0}]\!]_{\mathcal{F}} = 0$
- $[\![s]\!]_{\mathcal{F}}(n) = n + 1$
- $[\![+]\!]_{\mathcal{F}}(m, n) = m + n$
- $[\![\times]\!]_{\mathcal{F}}(m, n) = m \times n$
- $[\![\leq]\!]_{\mathcal{R}} = \{(m, n) \mid m, n \in \mathbb{N}, m \leq n\}$

Then $\mathcal{M} \models \forall x.\exists y.s(x) \leq y$*.*

Lemma 9. *Let* $\mathcal{M}$ *be a structure for some signature* Σ*,* φ *a formula, and* σ*,* σ' *variable assignments that agree on* $\mathrm{FV}(\varphi)$*. Then* $\mathcal{M}, \sigma \models \varphi$ *iff* $\mathcal{M}, \sigma' \models \varphi$*.*

Corollary 10. *The interpretation of a closed formula is independent of variable assignments.*

Lemma 11. *Alpha equivalent formulas evaluate to the same truth value.*

This shows that semantically there is no way to distinguish between alpha equivalent formulas. From here on, we will consider alpha equivalent formulas to be different ways of writing the same formula; substitution is then always defined.

Lemma 12. *The following equivalences hold in first-order logic:*

1. $(\forall x.\varphi) \Leftrightarrow \neg(\exists x.\neg\varphi)$
2. $(\forall x.\varphi \wedge \psi) \Leftrightarrow (\forall x.\varphi) \wedge (\forall x.\psi)$
3. $(\exists x.\varphi \vee \psi) \Leftrightarrow (\exists x.\varphi) \vee (\exists x.\psi)$
4. $(\forall x.\forall y.\varphi) \Leftrightarrow (\forall y.\forall x.\varphi)$
5. $(\exists x.\exists y.\varphi) \Leftrightarrow (\exists y.\exists x.\varphi)$
6. $(\exists x.\forall y.\varphi) \Rightarrow (\forall y.\exists x.\varphi)$*, but* not *vice versa*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P, Q \vdash P \qquad P, Q \vdash Q}{P, Q \vdash P \wedge Q}\,(\wedge\mathrm{R})}{P \vdash P \wedge Q, \neg Q}\,(\neg\mathrm{R})}{\vdash P \wedge Q, \neg P, \neg Q}\,(\neg\mathrm{R})}{\neg(P \wedge Q) \vdash \neg P, \neg Q}\,(\neg\mathrm{L})}{\neg(P \wedge Q) \vdash \neg P \vee \neg Q}\,(\vee\mathrm{R})$$

Figure 2: An example derivation in sequent calculus

In general we have neither $(\forall x.\varphi \vee \psi) \Leftrightarrow (\forall x.\varphi) \vee (\forall x.\psi)$ nor $(\exists x.\varphi \wedge \psi) \Leftrightarrow (\exists x.\varphi) \wedge (\exists x.\psi)$.
However, if $x \notin \mathrm{FV}(\varphi)$ *we have:*

1. $(\forall x.\varphi \vee \psi) \Leftrightarrow \varphi \vee (\forall x.\psi)$
2. $(\exists x.\varphi \wedge \psi) \Leftrightarrow \varphi \wedge (\exists x.\psi)$

While the agreement lemma for first-order logic looks very similar to its propositional equivalent, there is no such thing as a truth table for first-order logic. In fact, we have the following result:

Theorem 13 (Undecidability of First-order Logic)**.** *It is, in general, undecidable for a formula* φ *of first-order logic whether it is valid or satisfiable.*

7 Sequent calculus

The semantics of first-order logic gives us a clear description of which formulas are true and which are false, but it is not very helpful with finding a proof of a true formula, or with deriving the truth of one formula from the truth of the other.

The *sequent calculus*, introduced by Gerhard Gentzen in 1933, is a system for finding proofs of formulas. It defines a notion of a *derivation*, usually written in tree form, that shows why a formula must be true, decomposing it further and further until we come to formulas that are "obviously" true.

An example of a derivation in sequent calculus is given in Figure 2. We will use it as an example to explain how derivations are built up.

The derivation consists of *sequents* of the form

$$\Gamma \vdash \Delta$$

where Γ and Δ are finite sets of first-order logic formulas.[2] This means in particular that the order of the formulas in Γ and Δ does not matter, and that we can duplicate formulas as needed. We write Γ, φ to mean $\Gamma \cup \{\varphi\}$, and the empty string to mean the empty set.

[2]The symbol $\vdash$ is called "turnstile".

$$(\neg\text{L})\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma, \neg\varphi \vdash \Delta} \qquad (\neg\text{R})\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash \neg\varphi, \Delta}$$

$$(\wedge\text{L})\ \frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \varphi \wedge \psi \vdash \Delta} \qquad (\wedge\text{R})\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta}$$

$$(\vee\text{L})\ \frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vee \psi \vdash \Delta} \qquad (\vee\text{R})\ \frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta}$$

$$(\to\text{L})\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \varphi \to \psi \vdash \Delta} \qquad (\to\text{R})\ \frac{\Gamma, \varphi \vdash \psi, \Delta}{\Gamma \vdash \varphi \to \psi, \Delta}$$

$$(\forall\text{L})\ \frac{\Gamma, \varphi[t/x] \vdash \Delta}{\Gamma, \forall x.\varphi \vdash \Delta} \qquad (\forall\text{R})\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \forall x.\varphi, \Delta}\ \text{if } x \notin \mathrm{FV}(\Gamma, \Delta)$$

$$(\exists\text{L})\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \exists x.\varphi \vdash \Delta}\ \text{if } x \notin \mathrm{FV}(\Gamma, \Delta) \qquad (\exists\text{R})\ \frac{\Gamma \vdash \varphi[t/x], \Delta}{\Gamma \vdash \exists x.\varphi, \Delta}$$

$$(\textsc{subst})\ \frac{s \doteq t, \Gamma[s/x] \vdash \Delta[s/x]}{s \doteq t, \Gamma[t/x] \vdash \Delta[t/x]} \qquad (\textsc{cut})\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \varphi \vdash \Delta}{\Gamma \vdash \Delta}$$

Figure 3: Inference rules of LK

Intuitively, we understand a sequent $\Gamma \vdash \Delta$ to assert that whenever *all* formulas in Γ hold, *one* formula in Δ must hold. This, however, is just an intuition, not a formal definition!

The leaves of the tree at the very top must all carry *basic sequents*. A basic sequent is either of the form

$$\Gamma, \varphi \vdash \varphi, \Delta$$

where φ is an arbitrary formula; or it is of the form

$$\Gamma, \bot \vdash \Delta$$

or it is of the form

$$\Gamma \vdash t \doteq t, \Delta$$

where t is an arbitrary term. In all cases, Γ and Δ may be arbitrary finite sets.

Any sequent that is not a basic sequent must appear below a horizontal line, with one or more sequents above the line. The sequents above the line are called *antecedents*, the one below the line is the *consequent*. We call such a construction a *derivation step*.

Every derivation step has a label to the left of the horizontal line, which indicates the *inference rule* of which this step is an instance. The derivation rules of LK, the sequent calculus we are using in these notes, are given in Figure 3. A derivation step must match the indicated rule, with concrete formulas substituted for φ and ψ and concrete sets of formulas substituted for Γ and Δ.

The bottom-most sequent in a derivation is called the *conclusion* of the derivation; alternatively we say that the derivation is a derivation *of* that sequent.

At first sight, the sheer number and variety of inference rules might seem quite bewildering. There is, however, a beautiful symmetry behind these rules. Note that for every connective except $\bot$ there are two inference rules, a so-called *left rule* where the connective occurs on the left hand side of the turnstile in the consequent, and a *right rule* where it occurs on the right hand side.

The rules for $\wedge$ and $\vee$ are very symmetric, but with left and right rules reversed, and the same is true for $\forall$ and $\exists$. Only two rules do not fit into this schema: the substitution rule for equality and the cut rule.

You should now be able to convince yourself that the example derivation in Figure 2 is a genuine derivation, built up from basic sequents and inference rules.

Notice that such a derivation can be built quite systematically, starting with the conclusion: in every step, we choose a non-atomic formula either on the left or on the right, and apply the corresponding left or right rule. In this way, we "grow" the derivation bottom-up until we are only left with basic sequents.

For propositional formulas, this process is purely mechanical. For first-order logic, when we want to apply rules ($\forall$L) or ($\exists$R) we have to "guess" the term t to use in the antecedents. But of course we cannot really hope for an automatic way of constructing derivations for first-order formulas, since first-order logic is undecidable.

Notice that the rules ($\forall$R) and ($\exists$L) can only be applied if their side conditions are fulfilled (we write $\mathrm{FV}(\Gamma, \Delta)$ to mean $\mathrm{FV}(\Gamma) \cup \mathrm{FV}(\Delta)$). To see why this is necessary, consider the following flawed "derivation":

$$(\to\mathrm{R})\ \dfrac{(\forall\mathrm{R})\ \dfrac{(\exists\mathrm{L})\ \dfrac{P(x) \vdash P(x)}{\exists x.P(x) \vdash P(x)}}{\exists x.P(x) \vdash \forall x.P(x)}}{\vdash (\exists x.P(x)) \to (\forall x.P(x))}$$

Here, the side condition of ($\exists$L) is violated, since in that derivation step we have $\Delta = \{P(x)\}$, so $x \in \mathrm{FV}(\Delta)$. If we try to apply ($\forall$R) first, we likewise cannot satisfy its precondition. Of course, the above formula should not be derivable, since it is not valid according to our semantics of first-order logic.

Our list of inference rules contains rules for $\neg$; these are convenient, but not strictly necessary: after all, $\neg\varphi$ is just an abbreviation for $\varphi \to \bot$, so we can replace any application of ($\neg$L) by an application of ($\to$L) like this:

$$(\to\mathrm{L})\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \bot \vdash \Delta}{\Gamma, \neg\varphi \vdash \Delta}$$

Notice that the right antecedent is a basic sequent, so we only have to find a derivation of the left antecedent, which is the same as the antecedent of the ($\neg$L) rule.

For ($\neg$R), things don't look so promising at first. We can try applying ($\to$R)

$$(\to\text{R})\ \frac{\Gamma, \varphi \vdash \bot, \Delta}{\Gamma \vdash \neg\varphi, \Delta}$$

but now we have an extra $\bot$ on the right hand side in the antecedent. Fortunately, it turns out that this does not matter:

Lemma 14 (Weakening)**.** *If there is a derivation for $\Gamma \vdash \Delta$, then there is also a derivation for $\Gamma, \Gamma' \vdash \Delta, \Delta'$ for any finite sets Γ' and Δ'.*

Proof. Assume we have a derivation D of $\Gamma \vdash \Delta$; we construct a derivation of $\Gamma, \Gamma' \vdash \Delta, \Delta'$ by induction on the number of derivation steps in D.

If there are no derivation steps in D, then $\Gamma \vdash \Delta$ must be a basic sequent. If it is of the form $\Gamma'', \varphi \vdash \Delta'', \varphi$, then $\Gamma, \Gamma' \vdash \Delta, \Delta'$ is of the form $\Gamma'', \varphi, \Gamma' \vdash \Delta'', \varphi, \Delta'$, which is also a basic sequent. If it is a basic sequent of one of the two other forms, we again see easily that $\Gamma, \Gamma' \vdash \Delta, \Delta'$ is also a basic sequent.

If D has derivation steps, we consider the conclusion of D. It must have been obtained by one of the inference rules, so we do a case analysis on which rule was used to perform the final derivation step. In every case, we can assume that the result holds for the antecedents, since they have been built up with fewer derivation steps.

As an example, assume the final derivation step was an application of $(\wedge\text{R})$. Then Δ is of the form $\varphi \wedge \psi, \Delta''$, and the antecedents are $\Gamma \vdash \varphi, \Delta''$ and $\Gamma \vdash \psi, \Delta''$. By induction hypothesis we can assume that there are derivations for $\Gamma, \Gamma' \vdash \varphi, \Delta'', \Delta'$ and $\Gamma, \Gamma' \vdash \psi, \Delta'', \Delta'$, so by applying $(\wedge\text{R})$ we can find a derivation of $\Gamma, \Gamma' \vdash \varphi \wedge \psi, \Delta'', \Delta'$ as desired.

The other cases are similar, except for $(\forall\text{R})$ and $(\exists\text{L})$, where we need to do some renaming to make sure the side conditions are not violated. □

Sometimes, weakening rules are added as explicit rules to the sequent calculus; by the previous lemma this does not change the power of the system:

$$(\text{WL})\ \frac{\Gamma \vdash \Delta}{\Gamma, \Gamma' \vdash \Delta} \qquad\qquad (\text{WR})\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \Delta'}$$

Example 11. *As an example of the use of* (SUBST)*, here is how we can derive symmetry of equality; we choose $\Gamma = \emptyset$, $\Delta = \{x \doteq s\}$ for a variable $x \notin \mathrm{FV}(s)$:*

$$(\text{SUBST})\ \frac{s \doteq t \vdash s \doteq s}{s \doteq t \vdash t \doteq s}$$

We have here made use of the fact that if $x \notin \mathrm{FV}(s)$, then $s[t/x] = s$, which is easy to prove by structural induction.

Definition 19. *We write $\Gamma \vdash_{\mathrm{LK}} \Delta$ to mean that there is a derivation according to the rules of* LK *with the conclusion $\Gamma \vdash \Delta$.*

If $\vdash_{\mathrm{LK}} \varphi$, then we call φ a theorem *of* LK*.*

Example 12. *The following are theorems of* LK*, for any formulas φ and ψ:*

$\neg\neg\varphi \to \varphi$, $\varphi \vee \neg\varphi$, $\neg(\varphi \wedge \psi) \leftrightarrow \neg\varphi \vee \neg\psi$, $\neg(\varphi \vee \psi) \leftrightarrow \neg\varphi \wedge \neg\psi$, $\varphi \to \psi \leftrightarrow \neg\varphi \vee \psi$, $((\varphi \to \psi) \to \varphi) \to \varphi$, $(\forall x.\varphi) \to (\exists x.\varphi)$

Theorem 15 (Soundness of LK)**.** *The system* LK *is sound: If* $\Gamma \vdash_{\mathrm{LK}} \varphi$ *then* $\Gamma \models \varphi$*. In particular, all theorems are valid.*

Theorem 16 (Consistency of LK)**.** *The system* LK *is consistent, i.e., there is a formula* φ *such that we do not have* $\vdash_{\mathrm{LK}} \varphi$*.*

Proof. Indeed, take $\bot$. If we could derive $\vdash_{\mathrm{LK}} \bot$, then by the soundness lemma $\models \bot$. But that is not the case. □

Theorem 17 (Completeness of LK)**.** *The system* LK *is complete: If* $\Gamma \models \varphi$ *then* $\Gamma \vdash_{\mathrm{LK}} \varphi$*, i.e., all valid formulas can be derived.*

The proof of this theorem is quite difficult.

8 Peano arithmetic

One of the most basic fields of mathematics is arithmetic, i.e., calculating with the natural numbers. Arithmetic can be captured as a first-order *theory*, that is, a set of first-order formulas, over the signature $\Sigma_A := \langle \mathcal{F}_A, \emptyset \rangle$, where $\mathcal{F}_A := \{\mathbf{0}/0, s/1, +/2, \times/2\}$. Intuitively, $\mathbf{0}$ represents the number zero, s the successor function, and $+$ and $\times$ are addition and multiplication, respectively.

We define the theory of Peano arithmetic T_A as the set containing the following formulas (and no others):

- $\forall x.s(x) \doteq s(y) \to x \doteq y$
- $\forall x.\neg(s(x) \doteq \mathbf{0})$
- $\forall x.x + \mathbf{0} \doteq x$
- $\forall x.\forall y.x + s(y) \doteq s(x + y)$
- $\forall x.x \times \mathbf{0} \doteq \mathbf{0}$
- $\forall x.\forall y.x \times s(y) \doteq (x \times y) + x$
- for any formula φ: $\varphi[\mathbf{0}/x] \to (\forall x.\varphi \to \varphi[s(x)/x]) \to (\forall x.\varphi)$

Note that in any model $M = \langle D, I \rangle$ of T_A the function $I(s)$ must be injective, and $I(\mathbf{0})$ is not in its range: this is ensured by the first two formulas. The third and fourth formula describe the behaviour of $+$ and $\times$, whereas the (infinite) set of formulas added to T_A by the last rule makes sure that M must validate the induction principle.

We can now formally prove the validity of many arithmetic laws by showing that they are entailed by T_A, for instance $T_A \models \forall x.(x \doteq \mathbf{0} \vee \exists p.(x \doteq s(p)))$, or $T_A \models \forall x.\forall y.(x + y \doteq y + x)$.

Note that we defined the set of relation symbols to be empty. But we can easily define commonly used arithmetic relations as abbreviations; for instance, $t_1 \leq t_2$ can be encoded as $\exists z.t_1 + z \doteq t_2$, where we choose $z \notin \mathrm{FV}(t_1) \cup \mathrm{FV}(t_2)$, and $t_1 < t_2$ is $(t_1 \leq t_2) \wedge \neg(t_1 \doteq t_2)$.

Our signature does not contain function symbols for other commonly used arithmetic operators like exponentiation or the factorial function. Of course, we could add these symbols to the signature and extend T_A with formulas describing their meaning. For example, for exponentiation we could add the formulas $\forall x.x^0 \doteq s(z)$ and $\forall x.\forall y.x^{s(y)} \doteq x^y \times x$, which would enable us to prove the usual laws of calculating with exponents.

Surprisingly, this is not necessary as the following theorem tells us:

Theorem 18. *Let $\circ$ be a binary operator symbol; let g and h be terms not containing $\circ$ such that $\mathrm{FV}(g) \subseteq \{x\}$ and $\mathrm{FV}(h) \subseteq \{x, y, z\}$. Let ψ_1 be the formula*

$$\forall x.x \circ 0 \doteq g$$

and ψ_2 the formula

$$\forall x.\forall y.x \circ s(y) \doteq h[x \circ y/z].$$

Let $T_{A,\circ}$ be $T_A \cup \{\psi_1, \psi_2\}$.

Then for every formula φ, possibly using $\circ$, we can find a formula φ', not using $\circ$, such that $T_{A,\circ} \models \varphi$ iff $T_A \models \varphi'$.

This theorem says that for a large class of arithmetic operators that can be given a recursive definition with terms g and h of the above form we do not actually have to add any additional formulas to T_A in order to be able to reason about them. Instead, we can simply rewrite any formula that uses them into a simpler form where the new operator does not occur any longer. In particular, this theorem applies to exponentiation: choose g to be $s(\mathbf{0})$ and h to be $z \times x$.

In the above formulation of the theorem, the definition of the function needs to have precisely one non-recursive argument, namely x. This is not crucial: the theorem can be generalised to an arbitrary number of non-recursive arguments. Indeed, one can show that *any* computable function can be encoded in Peano Arithmetic. Carefully stating and proving this result is beyond the scope of this course, however.

9 Limits of first-order logic

While the previous section has shown that first-order logic is very powerful and can indeed serve as a basis for formalising much of mathematics, it is, in other respects, surprisingly weak. This weakness stems from the following result, which seems rather technical at first sight:

Theorem 19 (Compactness Theorem)**.** *First-order logic is* compact *in the following sense: A set Γ of formulas is satisfiable iff every finite subset Γ' of Γ is satisfiable.*

This theorem can be used to show that many concepts are not formalisable in first-order logic.

Example 13. *There is no first-order logic formula φ such that $\mathcal{M} \models \varphi$ iff the domain D of $\mathcal{M}$ is finite.*

Indeed, assume such a φ were given. For any natural number n we can define a formula λ_n such that $\mathcal{M}$ is a model for λ_n iff its domain has at least n elements; for instance, we can choose λ_3 to be

$$\exists x_1.\exists x_2.\exists x_3.(\neg x_1 \doteq x_2 \wedge \neg x_1 \doteq x_3 \wedge \neg x_2 \doteq x_3).$$

Let Λ be the set of all formulas λ_n.

Clearly, if $\mathcal{M} \models \Lambda$ then the domain of $\mathcal{M}$ must be infinite, so $\Lambda \cup \{\varphi\}$ is unsatisfiable.

On the other hand, any finite subset Λ' of Λ is satisfiable over a finite domain, for there is a maximal natural number m such that $\lambda_m \in \Lambda'$, but no $\lambda_{m'} \in \Lambda'$ for $m' > m$, so that any structure with a domain of size at least m' is a model for Λ'.

Consider now the set $\Lambda \cup \{\varphi\}$. Any finite subset of this set either is a finite subset of Λ, or it is a finite subset of Λ together with φ; in either case, it is satisfiable. But then by the compactness theorem $\Lambda \cup \{\varphi\}$ would have to be satisfiable, contradicting our assumption. This shows that such a φ cannot exist.

Example 14. *Let a signature Σ with a binary relation symbol R be given. Then there is no first-order formula R^* with two free variables x and y such that for any structure $\mathcal{M}$ we have $\mathcal{M}, [x := c, y := d] \models R^*$ iff $(c, d) \in [\![R]\!]^*$. Put more succinctly, there is no formula R^* that expresses the reflexive transitive closure of a relation symbol R.*

The proof goes as above, but considering a set built up from formulas $\overline{r}^{\leq n}$ expressing that y cannot be reached from x under R in n or less steps.

Of course, we *can* express the reflexive transitive closure of R by an infinite set of formulas, as we did for the induction principle of arithmetic above.