Luca Mauri

Lecture notes in

Logic and algebra

Academic year 2023-2024

Contents

Introduction

In these notes we are interested in logic as a tool to formalize and analyze reasoning. Although this analysis is done at different levels which depend on the nature of the logic used, we can roughly identify three main steps.

- Syntax. In order to analyse reasoning, a logic should have a language in which the reasoning can be formalized. The construction of this language and the manipulations that can be performed on it form the object of the *syntax* of the logic. It should be clear that the nature of the language used determines which types of reasoning the logic will be able to examine. The construction of a language starts with an alphabet and with rules to construct more complex expressions, the formulas, which are the main constituents of the language.

- Semantics. Having constructed formulas, we have to assign rules that determine when a formula is true or false, that is to assign a meaning to formulas. We also have to provide rules that tell us when a formula, the conclusion, can be regarded as a legitimate consequence of a set of other formulas, the premises. All this is the scope of *semantics*, which can be regarded as the core of every logic.

- Calculi. We also want to determine reasoning schemes that would allow us to start from premises we are willing to accept as correct and infer the correctness of more complex conclusions, thus validating a reasoning without checking truth or falsity of its formulas. These reasoning schemes are called logical *calculi* and a single logic can have many of them. We want a calculus to be coherent with semantics, in the sense that the calculus should validate exactly those reasoning which are correct from the semantical viewpoint.

1 Propositional logic

1.1 Overview

Propositional logic is an elementary tool which serves as a foundation for logics with a more complex and analytic structure.
It can be used to formalize and examine the correctness of reasonings constructed from only two types of entities: variables and connectives. Variables represent elementary assertions which can only be true or false; they are atomic in nature, in the sense that they can not be further analyzed or decomposed: different statements are represented by different variables and are treated as completely unrelated entities, even if they have some common or related content. Connectives are partly inspired by natural languages, partly imported from mathematics and computer science and are used to assemble variables into more complex constructs called formulas. The rules used in the construction of these formulas and the study of their formal properties form the *syntax* of the logic.
Once the reasoning has been converted into formulas, this abstracted form of reasoning can be analyzed for correctness giving a meaning to formulas. In propositional logic this meaning is reduced to the assertion that a particular formula is either true or false. The process of assigning a meaning to formulas is the *semantics* of the logic. By examining the relation between truth values of the formulas represnting the premises and the conclusions of the reasoning, we decide whether the abstract reasoning can be considered as a correct one.
Finally, we develop purely formal rules to decide whether a reasoning is correct, independently from but coherently with the semantical analysis, in a pure mechanical way. A set of rules that allows to decide about the correctness of a reasoning without looking at its meaning is called a logical *calculus*.
Syntax, semantics and calculi are the aspects of propositional logic that will be examined in this chapter.

1.2 Syntax

We start by introducing the languages that propositional logic uses to analyse reasoning. We do this by first defining what a propositional alphabet is, how the symbols of the alphabet are used to construct formulas, which are the elements of the language, and how these formulas can be manipulated.

□ **Propositional alphabets.** Every language starts with the description of an alphabet and of the rules by which the elements of the alphabet are used to construct more complex expressions. In the case of propositional logic, the possible alphabets have a very simple structure.

Definition 1.1. A *propositional alphabet* is assigned by three disjoint sets of symbols:

1. an arbitrary set V of *propositional variables*;
2. a finite set C of *propositional connectives*, together with an *arity* function $C \to \mathbf{N}$;
3. the set of *auxiliary simbols* consisting of the two parentheses (and) and of the comma.

None of the sets in the definition has a particular meaning at this point: they consists of purely formal symbols whose meaning will emerge when we will discuss semantics. Nevertheless, we can provide some intuition behind the definition. Variables are the symbols we use to reveal the structure of a reasoning and abstract its form. Symbols like x, y and z are common choices for variables; however, we might be in a situation in which a large number of variables is needed to formalize an argument; for this reason it is often more convenient to use as variables the elements of a set of the form $\{x_0, x_1, x_2, ...\}$ that uses a single letter x with indices from some ordinal number, finite or infinite, thus allowing for both a finite and an infinite supply of variables. In any case, the choice of variables is arbitrary, in the sense it it is only determined by the type of reasonings we want to examine.

Example 1.2 Consider the following argument in which n is a fixed integer.

> *"If n is a square, then if it is dividible by 2 it is also divisibe by 4. Therefore, if n is divisible by 2 and not by 4, it is not a square."*

A cursory examination reveals that the whole argument is constructed from three elementary assertions, which we call *atomic* because we do not wish to decompose them any further. We then introduce variables x, y and z to represent these atomic assertions.

$$x = n \text{ is a square}, \qquad y = n \text{ is divisible by } 2, \qquad z = n \text{ is divisible by } 4.$$

Replacing the atomic sentences with the corresponding variables in theoriginal reasoning yields, after minor grammar adjustments that do not alter the intended meaning, the following form, with added parentheses to clarify the hierarchy of the assertions.

> *"If x, then (if y then z). Therefore, if (y and (not z)) then (not x)."*

This form has the advantage of exposing the structure of the reasoning without distracting details. It also shows that the original reasoning is a special instance of something more general. In fact we could replace x, y and z with an arbitrary set of atomic sentences and obtain another reasoning of the same form. It is in this sense that x, y and z are *variables*: we use them as placeholder to abstract the structure of the reasoning, but they are not bound to any particular sentence; they can, in fact, vary over the realm of all atomic sentences and thus produce reasonings of the same form. This form of generalization implicitly introduces a problem, because we are not anymore examining the original argument, but rather a more general form of argument of which the given one is an instance. The conclusion we may draw from the more general argument will certainly apply to the more specidifc case, though there may be properties of the specific case that are not captured by the generalizarion.

Example 1.3 As an example to understand the atomic nature of propositional variables and how it limits the analytic possibilities of propositional logic, consider the following argument.

> *"Every integer divisible by 4 is also divisible by 2. The integer 12 is divisible by 4. Therefore 12 is divisible by 2."*

The reasoning is correct by any reasonable standard. However, when we try to replace the atomic statements with variables we are faced with the problem that each of the three assertions in the reasoning is atomic in nature and they are all different. We are thus forced to introduce as variables

$$\begin{aligned} x &= \text{every integer divisible by 4 is divisible by 2,} \\ y &= \text{12 is divisible by 4,} \\ z &= \text{12 is divisible by 2} \end{aligned}$$

and when we make this substitution the reasoning assumes the form

> *"x. y. Therefore z."*

Now this is a very general form of reasoning and we can not expect this to be true, since there is no relation between x, y and z. Thus, in the process of generalizing using variables, we have lost the capability of inferring the correctness of our original reasoning.

Propositional connectives are used to create dependency relations among expressions, much like conjunctions do for sentences in natural languages. In fact, many of the connectives used in propositional logic are inspired by those of natural languages, although not necessarily with the same meaning. The table below shows a fairly comprehensive list of propositional connectives.

Symbol	Name	Reads	Arity
$\top$	truth	true	0
$\bot$	falsehood	false	0
$\neg$	negation	not	1
$\wedge$	conjunction	and	2
$\vee$	disjunction	or	2
$\rightarrow$	implication	implies	2
$\leftrightarrow$	biimplication	if and only if	2
$\veebar$	exclusive disjunction	xor	2
$\uparrow$	negated conjunction	nand	2
$\downarrow$	negated disjunction	nor	2

(1.1)

We will examine the precise meaning of these connectives in the context of semantics, where we will also compare their meaning with that of the corresponding connectives of natural languages. Informally, though, some of the connectives in the previous table, namely negation, conjunction and disjunction, are derived from natural languages, with $\neg x$ representing the negation '*not x*', $x \wedge y$ corresponding to the coordinative conjunction '*x and y*' and $x \vee y$ to the disjuction '*x or y*'. Implication and biimplications are imports from mathematics: the implication $x \rightarrow y$ is used to abstract the statement '*if x then y*' typical of mathematical theorems, whereas the biimplication $x \leftrightarrow y$ represent the statement '*x if and only if y*' used to state equivalent conditions. Exclusive disjunction, negated conjunction and negated disjunction are more common in computer science where they are used to describe bit operations. Truth and falsehood can be understood as representations of absolute truth and falsity, as opposed to the nature of variables which can assume both truth values.
The arity of the connective will also be discussed when defining formulas. Intuitively, however, one can understand arity as the number of expressions the connective applies to: we negate one assertion at a time and take the conjunction or disjunction of two assertions. We therefore say that negation is a connective of arity 1 or a *unary* connective and that conjunction and disjunction have arity 2 or that they are *binary* connectives. It may seem surprising that no connectives of arity greater than 2 are listed in the table. There are two reasons for this fact: the first is that operations with more than two arguments are fairly rare in mathematics and likewise in natural languages; the second is that we will prove later on that arbitrary connectives of any arity are definable using those in the table. In fact, most of the connectives of the table are definable in terms of few of them, so that not all of them are actually necessary. The choice of the set of connectives to use in a propositional language has to balance the fact that a small number of connectives simplifies proofs, but can obscure the intended meaning of formulas. We will adopt an intermediate stance and use

$$C = \{\top, \bot, \neg, \wedge, \vee, \rightarrow, \leftrightarrow\} \tag{1.2}$$

as our set of standard connectives unless otherwise specified.

Example 1.4 Although the use of connectives is intertwined with the definition of formulas which has not been provided yet, we can informally see how connectives intervene in the formalization of reasoning. Returning to example 1.2 and to the first level of abstraction of the reasoning

"If x, then (if y then z). Therefore, if (y and (not z)) then (not x)"

we can replace natural connectives with propositional connectives as follows. We have already hinted at the fact that negation and conjunction from the natural language should be interpreted by $\neg$ and $\wedge$. Expressions like '*if x then y*' will be interpreted, as is usually done in mathematics, as '*x implies y*' using implication. As a result we can rewrite the reasoning in the form

"$x \rightarrow (y \rightarrow z)$. Therefore, $(y \wedge \neg z) \rightarrow \neg x$."

Note that we have not translated '*therefore*' with a connective. This is because connectives establish relations between parts of the same sentence; instead, '*therefore*' establishes a hierarchy between different sentences. We will see later on that it will be translated with the metalinguistic notion of consequence, represented by the symbol $\vDash$. Thus, the final translation of the reasoning into propositional logic reads

$$x \to (y \to z) \vDash (y \wedge \neg z) \to \neg x.$$

As to auxiliary symbols, they could be dispensed with a careful although quite unnatural choice of notation, known as polish or revers polish notation; this, however, would be at the expense of clarity. For this reason we will show later on how to eliminate commas, but we will retain parentheses. We will, however, provide rules to reduce their number to a minimum.

□ **Formulas.** Once a propositional alphabet has been chosen, we use its symbols to form more complex constructs, much like words in a natural language. As in a natural language not all sequences of letters are meaningful, though, so not all sequences of symbols from our alphabet will be useful. We therefore provide rules to isolate those which are of actual interest.

DEFINITION 1.5. The *formulas* generated by a propositional alphabet A are the sequences of symbols obtained by applying a finite number of times the following formation rules.

1. Every variable is a formula.
2. If c is a connective of arity n and $\varphi_1, \dots, \varphi_n$ are n formulas, then so is $c(\varphi_1, \dots, \varphi_n)$.

The set L of all formulas is the *propositional language* generated by A.

Although the definition describes formulas precisely, the notation it introduces tends to be cumbersome. Before giving specific examples, we therefore introduce some rules to simplify the notation. In the following discussion we will use the symbol $\Box$ to indicate a generic connective.

1. Connectives of arity 0 have a special nature. A connective c of arity n can be regarded as a function $c : L^n \to L$ associating to every n-tuple of formulas $(\varphi_1, \dots, \varphi_n) \in L^n$ the formula $c(\varphi_1, \dots, \varphi_n)$. In particular, since $L^0 = 1$ is the set with a single element and a function $1 \to L$ can be identified with its only value, a connective of arity zero can also be regarded as a formula. This agrees with the fact that if $\Box$ has arity 0, then $\Box()$ is a formula according to rule 2. We will therefore identify the formula $\Box()$ defined by a connective of arity 0 with the connective $\Box$ itself.
2. If $\Box$ is a connective of arity 1 we will replace $\Box(\varphi)$ with $(\Box\varphi)$. Note that the outermost parentheses are needed if we are further applying connectives to the formula.
3. If $\Box$ is a connective of arity 2 we use infix notation and substitute $\Box(\varphi_1, \varphi_2)$ with $(\varphi_1 \Box \varphi_2)$ as is customary with algebraic operations. Again, outermost parenthese are needed if further connectives are applied.

These rules, summarized in table 1.1 completely remove the necessity of commas in the language.

ARITY	STANDARD NOTATION	SIMPLIFIED NOTATION
0	$\Box()$	$\Box$
1	$\Box(\varphi)$	$(\Box\varphi)$
2	$\Box(\varphi_1, \varphi_2)$	$(\varphi_1 \Box \varphi_2)$

TABLE 1.1

We can also reduce the number of parentheses by eliminating the outermost parentheses in a formula and by assigning a priority to connectives, as shown in table 1.2. The rule is that a pair of parentheses can be omitted if the priority of the connective inside the parentheses is higher than the one of the connective outside. Connectives with the same priority will be applied left to right.

PRIORITY	CONNECTIVES
High	$\top, \bot$
¦	$\neg$
↓	$\wedge, \vee$
Low	$\to, \leftrightarrow$

TABLE 1.2

Note, however, that we will rarely remove all possible occurrences of parentheses, as keeping some of them improves readability of formulas. We will also retain parentheses in proofs when their use prevents ambiguity.
Variables and formulas defined by connectives of arity 0 are called *atomic formlas*.

Example 1.6 Let L be the propositional language generated by the set of variables $\{x, y\}$ and our standard set of connectives. We will prove that the sequence of symbols

$$(x \wedge \neg y \to x) \leftrightarrow \top \tag{1.3}$$

abbreviated with φ, is a formula of L. A first way to do this is to construct a *formation sequence* for φ: this is a finite sequence $(\varphi_1, \dots, \varphi_n)$ of formulas which are either atomic or obtained by applying one of the other formation rules to previously occurring formulas in the sequence, and such that $\varphi_n = \varphi$. One such formation sequence is given in table 1.3 below — the sequence is written vertically there.

STEP	FORMULA	RULE
1	x	atomic formula
2	y	atomic formula
3	$\neg y$	2, negation
4	$x \wedge \neg y$	1, 3, conjunction
5	$x \wedge \neg y \to x$	4, 1, implication
6	$\top$	atomic formula
7	$(x \wedge \neg y \to x) \leftrightarrow \top$	5, 6, biimplication

TABLE 1.3

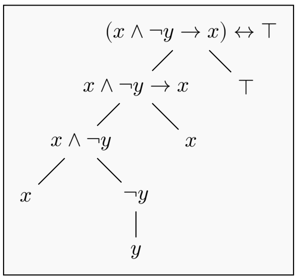


FIGURE 1.1

A second possibility is to construct a *tree* for φ as in figure 1.1. We can either construct the *formation tree* starting with atomic formulas at the leaves and then proceed upwards applying the formation rules to already constructed nodes, till we reach the root of the tree on top. Or we can construct the *parsing tree* starting from the root on top and then descend along the nodes using the formation rules till we arrive at atomic formulas in the leaves of the tree. Note that the outermost level of parentheses in the formula is not really needed, as the biimplication has the same priority of the implication, and since the latter occurs to the left it must be applied first; it has been preserved to impreve legibility.

Example 1.7 Providing an example of a sequence of symbols from a propositional alphabet which is not a formula is harder. As an example, we will show that if L is the propositional language generated by the variable x with the standard set of connectives, then the sequence of symbols $s := (\wedge x)$ is not a formula — we have retained all parentheses, here. To see this, let $M = L \setminus \{s\}$ and observe that $M \subseteq L$ by construction. Next, observe the following facts.

1. Every variable x is in M. In fact, $x \in L$ and it does not contain the symbol $\wedge$, hence $x \neq s$ and therefore $x \in M$.
2. If c is a connective of arity n and $\varphi_1, \dots, \varphi_n \in M$, then $c(\varphi_1, \dots, c_n) \in M$. This is because the φ_i's are in L and hence $c(\varphi_1, \dots, c_n) \in L$ but is different from s. This is clear if c is any connective different from $\wedge$, as s does not contain other connectives. If $c = \wedge$, then $(\varphi_1 \wedge \varphi_2) \neq (\wedge x)$ for otherwise $\varphi_1 = \varnothing$ and $\varphi_2 = x$ as there is only one level of parentheses in s; this contradicts the fact that every formula has positive length by the formation rules.

Since M contains all variables and is closed under the formation rules for formulas, it contains all the formulas; therefore $L \subseteq M$ and hence $L = M$. Thus, $s \notin M = L$ and s is not a formula.

□ **Structural induction and structural recursion.** The construction of the propositional language from a set of variables and connectives is a special case of a more general construction, the term algebra, which will be discussed in more detail in the chapter on algebraic theories. The properties of this construction, however, provide useful tools to work with propositional languages. For a first example, suppose we wish to prove that all formulas of a propositional language L have a property P. We write $\varphi \in P$ to indicate that φ has the property P, identifying the property with the set of formulas with property. With this convention, the assertion $P = L$ means that every formula has the property P.

Proposition 1.8. (Structural induction) If L is a propositional language and P is a property of formulas of L, then $P = L$ if and only if the following two conditions are satisfied.

1. $x \in P$ for every variable $x \in L$.
2. If c is a connective of arity n and $\varphi_1, \dots, \varphi_n \in P$, then $c(\varphi_1, \dots, \varphi_n) \in P$.

Proof. Necessity is clear, for L satisfies conditions 1 and 2 by definition. For sufficiency we will use properties that will be justified in chapter 4; we refer to diagram 1.4. If L is the propositional language generated by the set of variables X and the set of connectives C, then L is the free C-algebra on X. Conditions 1 and 2 above show that P is also a C-algebra for the tautological interpretation of the connectives and show that $X \subseteq P$; therefore it is a subalgebra of L and the inclusion $i : P \to L$ is a morphism of C-algebras under X. On the other hand, since L is the free C-algebra on X, there exists a unique morphism of C-algebras h under X. But then both $i \circ h$ and the identity on L are L-morphisms under X and by freeness they must be equal, $i \circ h = 1_L$. This implies that i is surjective and therefore that $P = L$. □

$$\begin{tikzcd} X \arrow[r] \arrow[dr] & L \arrow[d, dashed, shift left, "h"] \\ & P \arrow[u, shift left, "i"] \end{tikzcd} \tag{1.4}$$

Example 1.9 (Formation sequences) As an application of proposition 1.8, we prove that every formula φ of a propositional language L admits a formation sequence; since the elements of a formation sequence are formulas by definition, this will prove that the formulas of L are precisely the sequences of symbols admitting a formation sequence. To prove the claim observe first that every variable x admits the formation sequence (x). Now suppose c is a connective of arity n and $\varphi_1, \dots, \varphi_n \in L$ admit formation sequences $s_1, \dots, s_n$. Then, if we write $+$ for concatenation of sequences,

$$s := s_1 + \dots + s_n + (c(\varphi_1, \dots, \varphi_n)) \tag{1.5}$$

is a formation sequence for $c(\varphi_1, \dots, \varphi_n)$. By proposition 1.8, every formula of L has therefore a formation sequence.

Example 1.10 (Uniqueness of decomposition) As a second example of application of structural induction, one can prove that every formula has a unique decomposition, in the sense that for any formula $\varphi \in L$ exactly one of the following conditions hold:

- φ is a variable;
- $\varphi = c(\varphi_1, \dots, \varphi_n)$ for a uniquely determined connective c of arity n and a uniquely determined sequence $(\varphi_1, \dots, \varphi_n) \in L^n$.

Call this property P. P is true for variables, for which only the first condition can be true. If now c is a connective of arity n and $(\varphi_1, \dots, \varphi_n) \in L^n$ is a sequence of formulas satisfying P, then $\varphi = c(\varphi_1, \dots, \varphi_n)$ satisfies only the second in a unique form.
Another way to prove the claim is to define formulas as sequences: every variable is a sequence (x) of length 1; for every connective c of arity n we replace $c(\varphi_1, \dots, \varphi_n)$ with the $n+1$-tuple $(c, \varphi_1, \dots, \varphi_n)$. Because sequences of length n are equal exactly when they have the same coordinates, this automatically implies uniqueness of decomposition.

Uniqueness of decomposition has some useful applications. First, we can make more precise the notion of parsing tree of a formula: given a formula φ, either φ is a variable and its tree is reduced to φ, or $\varphi = c(\varphi_1, \dots, \varphi_n)$, in which case we can write φ at the root of the tree and its immediate descendants are the uniquely determined formulas $\varphi_1, \dots, \varphi_n$; one then repeats the arguments for the φ_i until atomic forulas are reaced.
A second application of the fact that a propsitional language is a free algebra is the construction of objects associated to formulas.

Proposition 1.11. (Definition by structural recursion) Assume L is a propositional language with a set X of variables and $f : X \to S$ is a function to an arbitrary set S. Suppose further that for every connective c of arity n we are given an operation $[\![c]\!] : S^n \to S$ of arity n on S. Then f extends uniquely to a function $f : L \to S$ satisfying the conditions

$$f(c(\varphi_1, \dots, \varphi_n)) = [\![c]\!]\,(f(\varphi_1), \dots, f(\varphi_n)). \tag{1.6}$$

Proof. The functions $[\![c]\!]$ provide S with a C-algebra structure. Because L is the free C-algebra on X, the function f_0 has a uunique extension to a morphism of $f : L \to S$ of C algebras; formula (1.6) is precisely the statement that f is a morphism. □

The situation described in the proposition is shown in the first diagram below, where we have written f_0 for the initial definition of f and j is the subset inclusion.

$$\begin{tikzcd}
X \arrow[r, hook, "j"] \arrow[dr, "f_0"'] & L \arrow[d, dashed, "f"] & L^n \arrow[l, "c"'] \arrow[d, dashed, "f^n"] & & X \arrow[r, hook, "j"] \arrow[dr, "\bar{f}_0 = (j, f_0)"'] & L \arrow[d, dashed, "\bar{f} = (i, f)"] & L^n \arrow[l, "c"'] \arrow[d, dashed, "\bar{f}^n = (i^n, f^n)"] \\
 & S & S^n \arrow[l, "{[\![c]\!]}"] & & & L \times S & L^n \times S^n \arrow[l, "\bar{c} = (c \circ p, {[\![c]\!]})"]
\end{tikzcd} \tag{1.7}$$

There is a variation of this property, shown on the second diagram, sometimes referred to as recursion with parameters which is often more useful.

Corollary 1.12. Assume L is a propositional language with a set X of variables and $f_0 : X \to S$ is a function to an arbitrary set S. Suppose further that for every connective c of arity n we are given a function $[\![c]\!] : L^n \times S^n \to S$. Then f_0 extends uniquely to a function $f : L \to S$ satisfying the conditions

$$f(c(\varphi_1, \dots, \varphi_n)) = [\![c]\!]\,(\varphi_1, \dots, \varphi_n, f(\varphi_1), \dots, f(\varphi_n)). \tag{1.8}$$

Proof. Define a function $\bar{f}_0 : X \to L \times S$ setting $\bar{f}_0(x) = (x, f_0(x))$ and for every connective c of arity n, define an n-ary operation $\bar{c}$ on $L \times S$ by the formula

$$\bar{c}(\varphi_1, \dots, \varphi_n, s_1, \dots, s_n) = (c(\varphi_1, \dots, \varphi_n), [\![c]\!]\,(\varphi_1, \dots, \varphi_n, s_1, \dots, s_n)). \tag{1.9}$$

By proposition 1.11 there exists a unique $\bar{f} : L \to L \times S$ making the second diagram commutative. If we break up $\bar{f}$ into its components $\bar{f} = (i, f)$, then the first component is the identity function on L. This can be proved by structural induction: for every variable x, commutativity of the left triangle yields $i(x) = j(x) = x$; assuming now that c is a connective of arity n and $\varphi_1, \dots, \varphi_n \in L$ such that $i(\varphi_i) = \varphi_i$, then by the commutativity of the right rectangle we have

$$i(c(\varphi_1, \dots, \varphi_n)) = c(i(\varphi_1), \dots, i(\varphi_n)) = c(\varphi_1, \dots, \varphi_n). \tag{1.10}$$

Thus i is the identity and looking at the second component of $\bar{f}$, commutativity of the right rectangle yields formula (1.8). □

Example 1.13 (FORMATION TREES) As a first application of theorem 1.11 we provide a precise definition of the formation tree of a formula. For our purposes, a tree is a finite partially ordered set T with minimum element, the root, and the property that for every $x \in T$ the set $(x) := \{y \in T : y \leq x\}$ is totally ordered and finite. The elements of T are the *nodes* of the tree. The cardinality of (x) is the *height* of x; the root has height 1; and if x has height n, the only element $y \in (x)$ of height $n-1$ is the *predecessor* of x and we say that x is a *successor* of y. Note that y can have more than one successor. We define an L-structure on the set S of finite trees with nodes in L. First, write $T\varphi$ to indicate a tree whose root is φ. If $\square$ is a connective of arity zero, the corresponding operation produces the tree whose only node is $\square$. If $\square$ is a connective of arity 1, the corresponding operation on S associates to a tree T with root φ a new tree obtained by adjoining $\square\varphi$ on top of the root. Finally, if $\square$ is a binary connective the induced operation takes the trees $T\varphi$ and $T\psi$ and adjoins a new root $\varphi\square\psi$ on top of them, as shown in the figure below.

$$\begin{tikzcd} \square\varphi \arrow[d, no head] \\ T\varphi \end{tikzcd} \qquad\qquad \begin{tikzcd} & \varphi\square\psi \arrow[dl, no head] \arrow[dr, no head] & \\ T\varphi & & T\psi \end{tikzcd} \tag{1.11}$$

To compute the formation tree in concrete examples it is more convenient to determine first a formation sequence and then to write the operations on the tree backwatds, starting with the last operation performed on the sequence. Thus, to construct the formation tree of $x \wedge \neg y \to \bot$ we start from a formation sequence $(x, y, x \wedge y, \bot, x \wedge y \to \bot)$ and construct the formation tree as follows

$$\begin{tikzcd} & x \wedge \neg y \to \bot \arrow[dl, no head] \arrow[dr, no head] & \\ T(x \wedge \neg y) & & T\bot \end{tikzcd} \quad \begin{tikzcd} & & x \wedge \neg y \to \bot \arrow[dl, no head] \arrow[dr, no head] & \\ & x \wedge \neg y \arrow[dl, no head] \arrow[dr, no head] & & \bot \\ Tx & & T(\neg y) & \end{tikzcd} \quad \begin{tikzcd} & & x \wedge \neg y \to \bot \arrow[dl, no head] \arrow[dr, no head] & \\ & x \wedge \neg y \arrow[dl, no head] \arrow[dr, no head] & & \bot \\ x & & \neg y \arrow[d, no head] & \\ & & Ty & \end{tikzcd} \quad \begin{tikzcd} & & x \wedge \neg y \to \bot \arrow[dl, no head] \arrow[dr, no head] & \\ & x \wedge \neg y \arrow[dl, no head] \arrow[dr, no head] & & \bot \\ x & & \neg y \arrow[d, no head] & \\ & & y & \end{tikzcd} \tag{1.12}$$

Example 1.14 (SUBFORMULAS) Informally, the subformulas of a formula φ are the nodes of its formation tree. A more explicit definition can be given using recursion as follows. The set $s(\varphi) \subseteq L$ of subformulas of φ is recursively defined by the following two conditions

1. If φ is a variable, $s(\varphi) = \{\varphi\}$.
2. If $\varphi = c(\varphi_1, \dots, \varphi_n)$, then $s(\varphi) = \bigcup_{i=1}^n s(\varphi_i) \cup \{\varphi\}$.

To see that this definition is in fact an instance of corollary 1.12, let $S = PL$ the set of subsets of formulas. Define first $s : V \to PL$ setting $s(x) = \{x\}$; that is, the only subformula of x is x itself; this is in fact the function $s_0 : X \to PL$. Now suppose c is a connective of arity n. Define

$$[\![c]\!]\,(\varphi_1, \dots, \varphi_n, F_1, \dots, F_n) = \bigcup_{i=1}^n F_i \cup \{c(\varphi_1, \dots, \varphi_n)\}\,. \tag{1.13}$$

By corollary 1.12, s extends uniquely to a function $s : L \to PL$. If $\varphi = c(\varphi_1, \dots, \varphi_n)$, then

$$s(\varphi) = s(c(\varphi_1, \dots, \varphi_n)) = [\![c]\!]\,(\varphi_1, \dots, \varphi_n, s(\varphi_1), \dots, s(\varphi_n)) \tag{1.14}$$

$$= \bigcup_{i=1}^n s(\varphi_i) \cup \{c(\varphi_1, \dots, \varphi_n)\} = \bigcup_{i=1}^n s(\varphi_i) \cup \{\varphi\} \tag{1.15}$$

which is exactly the definition given above. Thus, the subformulas of φ are φ itself together with all subformulas of the φ_i. For a concrete example, take the formula $\varphi = x \wedge \neg y \to \bot$ of example 1.13. We then have

$$s(\varphi) = \{x \wedge \neg y \to \bot\} \cup s(x \wedge \neg y) \cup s(\bot) \tag{1.16}$$

$$= \{x \wedge \neg y \to \bot,\ x \wedge \neg y,\ \bot\} \cup s(x) \cup s(\neg y) \tag{1.17}$$

$$= \{x \wedge \neg y \to \bot,\ x \wedge \neg y,\ \bot,\ x,\ \neg y\} \cup s(y) \tag{1.18}$$

$$= \{x \wedge \neg y \to \bot,\ x \wedge \neg y,\ \bot,\ x,\ \neg y,\ y\}\,. \tag{1.19}$$

It is immediate to verify that the subformulas obtained are exactly those found in the node of the formation tree of example 1.13.

Example 1.15 (DEGREE) Structural recursion can be used to assign a *degree* to every formula. First define $d : X \to \mathbf{N}$ setting $d(x) = 0$ for every variable. Next, for every connective c of arity n we define $[\![c]\!] : \mathbf{N}^n \to \mathbf{N}$ by the formula

$$[\![c]\!]\,(a_1, \dots, a_n) = 1 + \sum_{i=1}^{n} a_i. \tag{1.20}$$

By the structural recursion theorem the initial function d extends uniquely to a *degree* function $d : L \to \mathbf{N}$ such that

$$d(c(\varphi_1, \dots, \varphi_n) = 1 + \sum_{i=1}^{n} d(\varphi_i)\,. \tag{1.21}$$

What this function does, is to count the number of connectives in a formula. The usefulness of this function is that it can be used in proofs by induction: to prove that all formulas have a property P one can prove that all formulas of degree zero satisfy P and that if all formulas of degree n satisfy P, so do those of degree $n + 1$. A variation of this is to prove that if a formula of degree n does not satisfy P, then there is one of degree less than n that does not satisfy P. For a concrete example, let $\varphi = x \wedge \neg y \to \neg x \vee y$. We have

$$\begin{aligned} d(\varphi) &= 1 + d(x \wedge \neg y) + d(\neg x \vee y) && (1.22)\\ &= 3 + d(x) + d(\neg y) + d(\neg x) + d(y) && (1.23)\\ &= 5 + d(y) + d(x) && (1.24)\\ &= 5. && (1.25) \end{aligned}$$

☐ **Substitution.** A second important tool to work on propositional languages is substitution, which consists of replacing parts of a formula with other formulas.

DEFINITION 1.16. Suppose $\varphi \in L$ is a formula, $x := (x_1, \dots, x_m)$ is a finite sequence of variables and $\psi := (\psi_1, \dots, \psi_m)$ a sequence of formulas of the same length. The *substitution* of ψ for x in φ is the formula $\varphi[\psi/x]$ recursively defined as follows.

- If φ is a variable, then $\varphi[\psi/x] = \psi_i$ in case $\varphi = x_i$ and φ otherwise.
- If $\varphi = c(\varphi_1, \dots, \varphi_n)$, then $\varphi[\psi/x] = c(\varphi_1[\psi/x], \dots, \varphi_n[\psi/x])$.

This is an instance of the definition by recursion theorem (proposition 1.11)

$$\begin{tikzcd} X \arrow[r, hook, "j"] \arrow[dr, "f_0"'] & L \arrow[d, dashed, "f = {[\psi/x]}"] & L^n \arrow[l, "c"'] \arrow[d, dashed, "f^n = {[\psi/x]}^n"] \\ & S & S^n \arrow[l, "{[\![c]\!]}"] \end{tikzcd} \tag{1.26}$$

where

$$f_0(x) = \begin{cases} \psi_i & \text{if } x = x_i \\ x & \text{otherwise} \end{cases} \qquad [\![c]\!]\,(\varphi_1, \dots, \varphi_n) = c(\varphi_1, \dots, \varphi_n)\,. \tag{1.27}$$

The extension of f_0 to L is the substitution function $f = [\psi/x]$. Whether or not the x_i's occur in φ, it is sometimes convenient to write $\varphi(x)$ or $\varphi(x_1, \dots, x_m)$ to emphasize the possible dependence of φ on the x_i's; when we use this notation, we write $\varphi(\psi)$ or $\varphi(\psi_1, \dots, \psi_m)$ for $\varphi[\psi/x]$.

Example 1.17 Consider the substitution

$$\sigma = [x \to y/x,\ y \to x/y]\,. \tag{1.28}$$

If we apply this subsitution to the formula $\varphi = x \to y$ we find

$$\begin{aligned}\varphi\sigma &= (x \to y)\,[x \to y/x,\ y \to x/y] && (1.29)\\ &= x[x \to y/x,\ y \to x/y] \to y[x \to y/x,\ y \to x/y] && (1.30)\\ &= (x \to y) \to (y \to x)\,. && (1.31)\end{aligned}$$

The main application of substitution is that in order to prove that certain formulas have a property, one can try ot prove that this property is shared by simpler formulas from which the original formulas are obtained by substitution and that the property is stable under substitution.

1.3 Semantics

The purpose of semantics is to assign a meaning to every formula $\varphi \in L$ and on the basis of this to decide which reasonings are correct. Here '*meaning*' is understood in a fairly restrictive sense: we will only determine when φ is true or false, and we want to do so in a way which is coherent with the formation rules for formulas. For example, if we want to use the negation connective to to encode the usual negation from natural languages, we expect $\neg\varphi$ to be false when φ is true and conversely.
There is another important issue. Variables are neither intrinsically true or false: as their name suggests, they can be used to represent statements which might have either thuth value. As a consequence, we have to expect that the truth value of a formula will depend on that of its variables and not necessarily be a fixed one. To give another example, if conjucnction should encode the usual meaning of natural language conjunction, we expect $x \wedge y$ to be true when x and y are both true and false otherwise.
As it turns out, there are two equivalent ways to assign meaning as discussed above to propositional formula: a global approach and a local one. We will examine both.

□ **The interpretation of formulas, global version.** We fix a propositional language L generated by a set C of connectives and a set $V = \{x_0, x_1, x_2, \ldots\}$ of propositional variables.

Definition 1.18. A *truth assignment* on L is a function $v : L \to \Omega$, where

$$\Omega := \{0, 1\} \tag{1.32}$$

is the set of *truth values*. If $v(\varphi) = 1$ we say that φ is true in the assignment v or that v *satisfies* φ; if $v(\varphi) = 0$ we say that φ is *false* in the assignment v or that v does not satisfy φ.

We also write $v \vDash \varphi$ instead of $v(\varphi) = 1$ and read this as 'v *satisfies* φ'; similarly, we write $v \nvDash \varphi$ when $v(\varphi) = 0$ and read this as 'v *does not satisfy* φ'. A random assignment of truth values to formulas will not serve any purpose though, because it does not take into account neither formation rules nor any intended meaning of connectives. Instead, the truth value of $c(\varphi_1, \ldots, \varphi_n)$ should depend on the truth value of the φ_i's and on the intended meaning of c. We can express this dependency using the definition by recursion theorem and in particular formula (1.6). For this, we need for every connective c of arity n an operation $c_\Omega : \Omega^n \to \Omega$ interpreting the connective. For the purpose of the following definition we agree that $0 < 1$ in Ω.

Definition 1.19. If c is a propositional connective of arity n, its associated *truth function* $c_\Omega : \Omega^n \to \Omega$ is defined by the following formulas.

1. Connectives of arity 0 are interpreted by constants.
$$\bot_\Omega = 0, \qquad \top_\Omega = 1. \tag{1.33}$$
2. The only unary connective, negation, is interpreted by the function
$$\neg_\Omega x = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x = 1. \end{cases} \tag{1.34}$$
3. For the binary connectives we set
$$x \underset{\Omega}{\wedge} y = \begin{cases} 1 & \text{if } x = y = 1 \\ 0 & \text{otherwise,} \end{cases} \qquad x \underset{\Omega}{\vee} y = \begin{cases} 0 & \text{if } x = y = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{1.35}$$
$$x \underset{\Omega}{\longrightarrow} y = \begin{cases} 1 & \text{if } x \le y \\ 0 & \text{otherwise,} \end{cases} \qquad x \underset{\Omega}{\longleftrightarrow} y = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases} \tag{1.36}$$

The definition of truth functions can be summarized with the table

x	y	$\top_\Omega$	$\bot_\Omega$	$\neg_\Omega x$	$x \underset{\Omega}{\wedge} y$	$x \underset{\Omega}{\vee} y$	$x \underset{\Omega}{\rightarrow} y$	$x \underset{\Omega}{\leftrightarrow} y$
0	0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	1	0
1	0	1	0	0	0	1	0	0
1	1	1	0	0	1	1	1	1

(1.37)

Note that we could define negation, conjunction and disjunction in a more compact form as
$$\underset{\Omega}{\neg} x = 1 - x, \qquad x \underset{\Omega}{\wedge} y = \min(x, y), \qquad x \underset{\Omega}{\vee} y = \max(x, y). \tag{1.38}$$

The definition assigns a meaning to the connectives. For example, formula (1.34) expresses the fact that the negation of a true statement is false and, conversely, the negation of a false statement is true, which is in agreement with the use of negation in natural languages. The desire to keep this agreement, at least to a reasonable extent, is justified by the necessity of using logic to formalize and validate reasonings which are often expressed semi-formally in natural languages. However, the situation is already problematic with conjunction. Propositional conjunction is comparable with the coordinative usage of the conjunction '*and*' from natural languages; however, conjunction in natural languages often has a temporal or even a causal meaning. For a famous example, consider the two phrases

'*John fell sick and went to the doctor*', '*John went to the doctor and fell sick.*'

For a purely coordinative conjunction, commuting the sentences in the phrase should have no effect on meaning, which is not what happens here. The situation is even more blurred for other connectives. Disjunction in natural languages usually comes in two forms: an inclusive one, as in the first sentence below, and an exclusive one, as in the second.

'*I could eat apple pie or cheesecake*', '*For one coin you can get a drink or a snack.*'

In the first case the assertion is true if you eat any of the two or both desserts; in the second, it would seem wrong if you could get both a drink and a snack for a single coin. In the case of natural languages it is the context that tells us which meaning is correct; in logic we have to make a choice once and for all. It is for

this reason that we use the disjunction symbol for inclusive disjunction of natural languages and reserve the exclusive disjunction symbol for the other case. We conclude with some remarks on implication. The sentence *"If you open the window I will get cold"* would be reasonably considered true if the window is open and you get cold and false if the window is open and you don't. However, if the window is closed the sentence is usually given no truth value in natural languages. In logic, however, we can not leave undetermined cases and the use of implication is modeled on mathematics; there, a sentence like *"If x is a square, then it is not prime"* can be regarded as a theorem and the only way a theorem is false is when the premises are true and the conclusion is false.
With true functions defined for connectives, we can now provide a precise definition of what a meaningful truth assignment for formulas should look like.

Definition 1.20. A *valuation* on L is a truth assignment $v : L \to \Omega$ which is compatible with truth functions in the sense that for every connective c of arity n and for every n-tuple of formulas we have

$$v(c(\varphi_1, \dots, \varphi_n)) = c_\Omega(v(\varphi_1), \dots, v(\varphi_n)). \tag{1.39}$$

Formula (1.39) can be expressed saying that a valuation v is a truth assignment making the following diagram commutative.

$$\begin{tikzcd} L^n \arrow[r, "c"] \arrow[d, "v^n"'] & L \arrow[d, "v"] \\ \Omega^n \arrow[r, "c_\Omega"'] & \Omega \end{tikzcd} \tag{1.40}$$

From theorem 1.11 we then immediately have

Proposition 1.21. Every function $V \to \Omega$ extends uniquely to a valuation $L \to \Omega$. □

Thus, a valuation is determined by its value on variables. In view of this property, we will identify the set of valuations on L with the set $\Omega^V = \{v : V \to \Omega\}$ of all functions from the set V of variables to Ω. Observe that Ω^V is infinite if V is; if, however, the cardinality of V is finite, say $|V| = n$, then there are 2^n different valuations.

Example 1.22 Let L be the language generated by the set of variables $V = \{x, y, z\}$ and by the standard set of connectives C, let $\varphi := x \vee \neg y \to x \wedge (z \leftrightarrow \bot)$ and let $v \in \Omega^V$ be the valuation defined by

$$v(x) = 1, \qquad v(y) = 0, \qquad v(x) = 0. \tag{1.41}$$

Then, using formula (1.39) and the table in (1.37), we have

$$\begin{aligned} v(\varphi) &= v(x \vee \neg y) \underset{\Omega}{\longrightarrow} v(x \wedge (z \leftrightarrow \bot)) && (1.42)\\ &= v(x) \vee_\Omega v(\neg y) \to_\Omega v(x) \wedge_\Omega v(z \leftrightarrow \bot)) && (1.43)\\ &= 1 \vee_\Omega (\neg_\Omega v(y)) \to_\Omega 1 \wedge_\Omega (v(z) \leftrightarrow_\Omega v(\bot)) && (1.44)\\ &= 1 \vee_\Omega (\neg_\Omega 0) \to_\Omega 1 \wedge_\Omega (0 \leftrightarrow_\Omega 0) && (1.45)\\ &= 1 \vee_\Omega 1 \to_\Omega 1 \wedge_\Omega 1 && (1.46)\\ &= 1 \to_\Omega 1 && (1.47)\\ &= 1 && (1.48) \end{aligned}$$

It is often more covenient, to evaluate $v(\varphi)$ to list its subformulas and evaluate them progressively in a table as shown below.

x	y	z	$\neg y$	$x \vee \neg y$	$\bot$	$z \leftrightarrow \bot$	$x \wedge (z \leftrightarrow \bot)$	$x \vee \neg y \to x \wedge (z \leftrightarrow \bot)$
1	0	0	1	1	0	1	1	1

(1.49)

An alternative to this is to write the truth value of a subformula under its outermost connective as shown below. The value $v(\varphi)$ is then read under the main connective.

x	$\vee$	$\neg$	y	$\to$	x	$\wedge$	$(z$	$\leftrightarrow$	$\bot)$
1	1	1	0	1	1	1	0	1	0

(1.50)

To summarize the situation, in the global approach one fixes a valuation by assigning its value on variables and this automatically assigns a truth value to all formulas.

□ **The interpretation of formulas, local version.** Although the global approach is easily described, one rarely needs to know the truth value of all formulas for a single valuation; often it is more important to know how the truth value of a single formula changes when the valuation does. This is precisely what is done in the local approach: we fix a formula φ and assign to it a function $[\![\varphi]\!] : \Omega^V \to \Omega$, known as its interpretation or truth table.
To fix notation, if $V = \{x_0, x_1, \dots\}$, we can think of a valuation $a \in \Omega^V$ as a sequence $a = (a_0, a_1, \dots)$ of elements of Ω, where $a_i := a(x_i)$ is the truth value that a assigns to the i-th variable.

Definition 1.23. The *interpretation* of a formula $\varphi \in L$ is a function $[\![\varphi]\!] : \Omega^V \to \Omega$ assigning to every valuation $a \in \Omega^V$ a truth value $[\![\varphi]\!](a) \in \Omega$, recursively defined by the following rules.

1. The variable x_i is interpreted by the projection on the i-th coordinate,
$$[\![x_i]\!](a) = a_i. \tag{1.51}$$
2. If $c \in C$ sia a connective of arity n and $(\varphi_1, \dots, \varphi_n)$ is a sequence of formulas, the interpretation of $c(\varphi_1, \dots, \varphi_n)$ is the function defined by the formula
$$[\![c(\varphi_1, \dots, \varphi_n)]\!](a) = c_\Omega([\![\varphi_1]\!](a), \dots, [\![\varphi_n]\!](a)). \tag{1.52}$$

To see that this definition is really recursive, write $[\Omega^V, \Omega]$ for the set of all functions $\Omega^V \to \Omega$. In the diagram below, $[\![\_]\!]_0$ is the function that assigns to every variable x_i the i-th projection $[\![x_i]\!]$ defined by (1.51).

$$\begin{tikzcd}
V \arrow[r, tail, "i"] \arrow[dr, "{[\![\_]\!]_0}"'] & L \arrow[d, dashed, "{[\![\_]\!]}"] & L^n \arrow[l, "c"'] \arrow[d, dashed, "{[\![\_]\!]^n}"] \\
& {[\Omega^V, \Omega]} & {[\Omega^V, \Omega]^n} \arrow[l, "{[\![c]\!]}"]
\end{tikzcd} \tag{1.53}$$

The second part of the definition (formula (1.52)) corresponds to the commutativity of the rectangle. The connective c defines an n-ary operation $[\![c]\!]$ on $[\Omega^V, \Omega]$ as follows: given n functions $(f_1, \dots, f_n) \in [\Omega^V, \Omega]$, $[\![c]\!](f_1, \dots, f_n) \in [\Omega^V, \Omega]$ is defined by the assignment

$$a \mapsto c_\Omega(f_1(a), \dots, f_n(a)) \tag{1.54}$$

obtained by composing with the truth function $c_\Omega : \Omega^n \to \Omega$ defined by c, as shown in the diagram below.

$$\begin{tikzcd}
\Omega^V \arrow[rr, dashed, "{[\![c]\!](f_1, \dots, f_n)}"] \arrow[dr, "{(f_1, \dots, f_n)}"'] & & \Omega \\
& \Omega^n \arrow[ur, "c_\Omega"'] &
\end{tikzcd} \tag{1.55}$$

By theorem 1.11 we have a unique extension of $[\![\_]\!]_0$ to a function $[\![\_]\!] : L \to [\Omega^V, \Omega]$ satisfying formula (1.52); this extension assigns to every φ its interpretation $[\![\varphi]\!]$.
The global and local methods of evaluating formulas are actually different aspects of the same construction. To see this, observe that given a formula $\varphi \in L$ and a valuation $a \in \Omega^V$, we have two ways of assigning a truth value to φ using a: we can use $a(\varphi)$ as in the global approach and $[\![\varphi]\!](a)$ as in the local approach. The two calculations yield the same same result though, because formulas (1.39) and (1.52) which recursively define the two formulas are identical, as both amount to the commutativity of the same diagram 1.40. Thus, we have an evaluation function

$$\begin{tikzcd} L \times \Omega^V \arrow[r, "e"] & \Omega \\ (\varphi, a) \arrow[r, mapsto] & a(\varphi) = [\![\varphi]\!](a) \end{tikzcd} \tag{1.56}$$

If we transpose e with repsect to the first variable we obtain a function $\Omega^V \to [L, \Omega]$ which to every valuation a associates the function $\varphi \mapsto a(\varphi)$, as in the global approach: we fix a valuation and see what values are assigned to all formulas. If, on the other hand, we transpose with respect to the second variable, we obtain a function $L \to [\Omega^v, \Omega]$ associating to every formula φ its interpretation $a \mapsto [\![\varphi]\!](a)$, as in the local approach: we fix a formula and see how its truth value changes with the valueation. Thus, the two approaches are different manifestations of the same concept.
Although equivalent, each approach has its own advantages; in particular, we can use the local approach to examine change of variables. Suppose we fix a set C of propositional connectives and let L_U be the propositional language generated by a set U of variables. If $V \supseteq U$ is a larger set of variables, we have a new language L_V. Every formula $\varphi_U \in L_U$ is also a formula $\varphi_V \in L_V$, hence $L_U \subseteq L_V$ and we call L_V an *expansion* of L_U; from the opposite perspective, we say that L_U is a *reduct* of L_V. Now observe that the inclusion $i : U \to V$ induces by composition a *restriction* $i^* : \Omega^V \to \Omega^U$ defined by the formula $i^*(a) = a \circ i$, which limits the assignment of truth values from the variables in V to variables in U. We then have a commutative diagram

$$\begin{tikzcd} \Omega^V \arrow[r, "{[\![\varphi_V]\!]}"] \arrow[d, "i^*"'] & \Omega \\ \Omega^U \arrow[ur, "{[\![\varphi_U]\!]}"'] & \end{tikzcd} \tag{1.57}$$

because the evaluation process is defined in the same way regardless of the number of variables. We can look at the same diagram in the opposite perspective: if $\varphi_V \in L_V$ only contains only variables of U, then φ_V comes from a unique φ_U and $[\![\varphi_V]\!]$ is an extension of $[\![\varphi_U]\!]$; the extra variables of V play no role in φ and its restriction to U is all we need to know. The reason for which this is important is that the recursive definition of formulas only allows a finite number of steps in the construction, so that in fact the number of variables occurring in φ is finite even when V is infinite. We can not really describe $[\![\varphi]\!]$ when V is infinite, but we can do that for a finite U. Let us make this precise.

Definition 1.24. If L is the propositional language generated by a set of variables V and a set of connectives C, the *support* of a formula $\varphi \in L$ is the subset $S \subseteq V$ of variables which occur as subformulas of φ.

The support of φ is also called the *set of variables of* φ. Since every formula is a string of finite length we immediately have the following result, which can also be proved as an exercise using induction.

Proposition 1.25. The support of every formula $\varphi \in L$ is finite. □

Now suppose $\varphi \in L$ and its support is $U = \{x_1, \dots, x_n\}$; we will write $\varphi(x_1, \dots, x_n)$ to indicate that we are looking at the formula in the reduct. Since $|U| = n$, then $|\Omega^U| = 2^n$ because given any $a \in \Omega^U$ we can choose $a_1 = a(x_1)$ in two different ways; for each of these choices, we have two choces for a_2, so that a_1 and a_2 can be chosen in 2^2 ways and so on. Now the domain of $[\![\varphi]\!]$ is finite and we can compute $[\![\varphi]\!]$ in a finite number of steps; this function is also called the *truth table* of φ.

Example 1.26 In the propositional language generated by the standard set of connectives and by the set of variables $\{x, y, z\}$ consider the formula

$$\varphi := (x \wedge y \to z) \to (x \vee y \to z) \tag{1.58}$$

To compute its interpretation as a truth table, we first determine all its subformulas, which we write in order of increasing complexity, starting with the variables, in the first row of the table below, and then construct progressively the truth functions of the subformulas. To do this, observe that the 3 variables produce $2^3 = 8$ valuation which we list by increasing lexicographic order in the rows of the table.

x	y	z	$x \wedge y$	$x \wedge y \to z$	$x \vee y$	$x \vee y \to z$	φ
0	0	0	0	1	0	1	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	1
1	0	0	0	1	1	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1

(1.59)

The table itself is computed column by column until we reach the final column of φ. More in detail, the three initial columns contain the values assigned to the variable by all possible valuation. Column 4 is the conjunction of the values in column 1 and 2 and therefore will only be true when the values in both columns are, that is for the last two valuations. Likewise, column 5 is the implication with antecedent in column 4 and consequent in column 3 and will be false only when the first is true and the second false, i.e. only for the valuation in row 7. The remaining columns are computed similarly. From the last column we now see that $[\![\varphi]\!]$ is false only for v_3 and v_5, the third and fifth valuations.
Note that the table we have constructed would be the same for every language whom φ may belong, regardless of the cardinality of the set of variables; what determines the table is the support of φ. Also note that the interpretation of truth table of φ, strictly speaking, is only given by the 3 column sof the variables and the column of φ; the intermediate columns are there only as intermediate stages of the computation.

□ **Truth and satisfiability.** Having defined what it means for a truth assignment to satisfy a formula, we now classify formulas according to the nature of their interpretation.

Definition 1.27. Given a formula $\varphi \in L$ and a valuation $a \in \Omega^V$, we say that φ is *true* on a if $[\![\varphi]\!](a) = 1$ and that φ is *false* on a if $[\![\varphi]\!](a) = 0$. We say that φ is

- *valid* if $[\![\varphi]\!](a) = 1$ for every valuation $a \in \Omega^V$,
- *satisfiable* if $[\![\varphi]\!](a) = 1$ for at least one valuation $a \in \Omega^V$,
- *unsatisfiable* if $[\![\varphi]\!](a) = 0$ for every valuation $a \in \Omega^V$.

There is quite a number of variations in terminology. If $[\![\varphi]\!](a) = 1$ we also say that a *satisfies* φ or that a is a *model* of φ and write $a \vDash \varphi$. Likewise, if $[\![\varphi]\!](a) = 0$ we also say that a does not satisfy φ or that a is not a model of φ and write $a \nvDash \varphi$. Thus, φ is valid if every valuation is a model of φ; it is satisfiable if at least one valuation is a model of φ; it is unsatisfiable if it has no models. We write $\vDash \varphi$ to indicate that φ is valid.
Valid formulas are also clled *tautologies* and unsatisfiable formulas *contradictions*. Note that every valid formula is automatically satisfiable; a formula which is satifiable but not valid is sometimes called a *contingency*.

Example 1.28 Consider the formulas

$$(x \vee \neg x \vee z) \to (y \wedge \neg y \wedge z) \qquad x \vee y \to x \wedge z, \qquad x \wedge y \to x \vee z. \tag{1.60}$$

The supports of the three formulas coincide, which allows us to collect their truth tables into a single one as follows.

x	y	z	$(x \vee \neg x \vee z) \to (y \wedge \neg y \wedge z)$	$x \vee y \to x \wedge z$	$x \wedge y \to x \vee z$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	0	1	1

(1.61)

The table shows that the first formula is unsatisfiable, the second is satisfiable though not valid and the third is valid.

The definition generalizes to arbitrary sets of formulas.

Definition 1.29. A valuation a *satisfies* a set of formulas $F \subseteq L$ if it satisfies all the formulas of F; in this case we write $a \vDash F$ and we also say that a is a *model* of F. We also stipulate that every valuation satisfies the empty set $\varnothing$. The set F is *satisfiable* if there exists at least a valuation satisfying it and *unsatisfiable* otherwise.

Thus, F is unsatisfiable if for every $a \in \Omega^V$ there is at least one $\varphi \in F$ such that $[\![\varphi]\!](a) = 0$. Observe that even if all formulas of F are satisfiable F need not be, because satisfiability of F means that there is a valuation satisfying all formulas of F, whereas in the case of a set of satisfiable formulas each formula might have its own valuation satisfying it. Certainly a satisfiable set consists of satisfiable formulas, though the converse is not true. Dually, if F contains even a single unsatisfiable formula it will be unsatisfiable, but it might be unsatisfiable even if all its formulas are satisfiable.
The fact that the empty set is satisfied by all valuations can be justified as follows: consider the function $\mathcal{P}L \to \mathcal{P}\Omega^V$ associating to every set of formulas $F \subseteq L$ the set of valuations satisfying F; if $F \subseteq G$ then every valuation satisfying G also satisfies F; in other words, the smaller the set F becomes, the larger the set of valuations satisfying it; since $\varnothing$ is the smallest set, we therefore assume that the set of valuations satisfying it is the largest, i.e. the whole Ω^V. From another perspective we could say that $a \vDash \varnothing$ because there is no $\varphi \in \varnothing$ such that $a \nvDash \varphi$.

Example 1.30 Consider the set of formulas

$$F = \{x \to y,\ y \to z,\ z \to x\}, \qquad G = \{x \leftrightarrow \neg y,\ y \leftrightarrow \neg z,\ z \leftrightarrow \neg x\}\,. \tag{1.62}$$

The truth tables for the two sets are given below.

x	y	z	$x \to y$	$y \to z$	$z \to x$
0	0	0	1	1	1
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

x	y	z	$x \leftrightarrow \neg y$	$y \leftrightarrow \neg z$	$z \leftrightarrow \neg x$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	0

(1.63)

F is satisfiable because v_1 satisfies all the formulas of F; in fact also v_8 does. Instead, G is unsatisfiable because no single valuation satisfies all the formulas of G. Note, nevertheless, that all formulas of G are individually satisfiable.

Given a set of formulas $F \subseteq L$ we may look at the set of valuations satisfying F and conversely, given a set of valuations $U \subseteq \Omega^X$ we may look at the set of formulas satisfied by U. We start to examine the situation by restricting our language to one generated by the set of connectives $C = \{\bot, \top, \neg, \wedge, \vee, \to\}$.

Definition 1.31. A set of formulas $F \subseteq L$ is *saturated* if it satisfies he following conditions.

1. $\top \in F$ and $\bot \notin F$.
2. $\varphi \in F$ if and only if $\neg\varphi \notin F$ for every formula φ.
3. $\varphi \wedge \psi \in F$ if and only if $\varphi \in F$ and $\psi \in F$
4. $\varphi \vee \psi \in F$ if and only if $\varphi \in F$ or $\psi \in F$.
5. $\varphi \to \psi \in F$ if and only if $\varphi \notin F$ or $\psi \in F$.

Saturated sets bear a close relationship with valuations.

Proposition 1.32. A set of formulas $F \subseteq L$ is saturated if and only if it is the set of formulas satisfied by a valuation $a \in \Omega^X$. In this case, a is the characteristic function of F.

In view of the proposition, we can therefore say that valid formulas are those in the intersection of all saturated sets and satisfiable formulas are those in the their union. Similarly a satisfiable set must be contained in a saturated set, though can be smaller than a saturated set.

□ **Equivalence.** The formulas φ and $\neg\neg\varphi$ have different length and this suffices to see that they are different; however, they have the same interpretation $[\![\varphi]\!] = [\![\neg\neg\varphi]\!]$ because they are true exactly in the same instances. This makes them indistingushable from the samentical viewpoint. The problem of determining when two formulas have the same interpretation is addressed by the notion of equivalence.

Definition 1.33. Two formulas $\varphi, \psi \in L$ are *equivalent* if they have the same same truth function, that is if for every valuation $a \in \Omega^V$ we have

$$[\![\varphi]\!]\,(a) = [\![\psi]\!]\,(a)\,. \tag{1.64}$$

We write $\varphi \equiv \psi$ when this happens.

The fact that equivalent formulas have the same truth function is the precise meaning of the fact that equivalent formulas are indistinguishable from the semantical viewpoint; in other words, equivalence behaves like equality if we are only interested in the meaning of formulas. Also observe that since there are only two truth values in propositional logic, φ and ψ are equivalent if $[\![\varphi]\!](a) = 1 \Leftrightarrow [\![\psi]\!](a) = 1$, i.e. if they have the same models. Thus. to prove that two formulas are equivalent, it suffices to compute their truth table.

Example 1.34 We wish to determine which of the following formulas are equivalent.

$$x \wedge y \to z, \qquad x \to (y \to z), \qquad (x \to y) \to z. \tag{1.65}$$

The table below contains the truth tables of the three formulas.

x	y	z	$x \wedge y \to z$	$x \to (y \to z)$	$(x \to y) \to z$
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1

(1.66)

The fact that columns 3 and 4 in the table are equal shows that the first two formulas are equivalent, whereas the fact that the last column is different shows that the third formula is not equivalent to the other two.

Observe that the concept of semantical equivalence belongs to the metalanguage and not to the object language. However, it can be translated into a stetement of the object language using biimplication.

Proposition 1.35. (Internalization of equivalence) Two formulas $\varphi, \psi \in L$ are equivalent if and only if their biimplication is a valid formula. In symbols,

$$\varphi \equiv \psi \quad \Leftrightarrow \quad \vDash \varphi \leftrightarrow \psi. \tag{1.67}$$

Proof. By definitions 1.23 and 1.19 we have that for any $a \in \Omega^V$

$$[\![\varphi \leftrightarrow \psi]\!](a) = [\![\varphi]\!](a) \underset{\Omega}{\longleftrightarrow} [\![\psi]\!](a) = \begin{cases} 1 & \text{if } [\![\varphi]\!](a) = [\![\psi]\!](a) \\ 0 & \text{if } [\![\varphi]\!](a) \neq [\![\psi]\!](a). \end{cases} \tag{1.68}$$

Therefore, we have a chain of equivalent statement

$$\begin{aligned} \varphi \equiv \psi &\Leftrightarrow \text{for all } a \in \Omega^V,\ [\![\varphi]\!](a) = [\![\psi]\!](a) & (1.69)\\ &\Leftrightarrow \text{for all } a \in \Omega^V,\ [\![\varphi \leftrightarrow \psi]\!](a) = 1 & (1.70)\\ &\Leftrightarrow \vDash \varphi \leftrightarrow \psi & (1.71) \end{aligned}$$

where the first equivalence follows from the definition of equivalence, the second from formula (1.68) and the last from the definition of valid formula. □

The simplest technique to construct equivalent formulas is to use substitution. To fix notation, suppose $\psi = (\psi_1, \dots, \psi_n)$ and $\psi' = (\psi'_1, \dots, \psi'_n)$ are sequence of fomulas of the same length n we write $\psi \equiv \psi'$ if $\psi_i \equiv \psi'_i$ for $1 \leq i \leq n$.

Lemma 1.36. (Substitution lemma) Assume $\varphi, \varphi' \in L$ are formulas, x is a sequence of variables of length n and $\psi, \psi' \in L^n$. Then the following holds.

1. If $\psi \equiv \psi'$ then $\varphi[\psi/x] \equiv \varphi[\psi'/x]$
2. If $\varphi \equiv \varphi'$ then $\varphi[\psi/x] \equiv \varphi'[\psi/x]$

As a consequence, if $\varphi \equiv \varphi'$ and $\psi \equiv \psi'$ then $\varphi[\psi/x] \equiv \varphi'[\psi'/x]$.

Proof. ① The proof is by structural induction on φ. If φ is a variable, we have two cases, depending on whether $\varphi = x_i$ occurs in x or $\varphi = x_j$ does not occur in x:

$$\varphi[\psi/x] = x_i[\psi/x] = \psi_i \equiv \psi' = x_i[\psi'/x] = \varphi[\psi'/x], \tag{1.72}$$

$$\varphi[\psi/x] = x_j[\psi/x] = x_j = x_j[\psi'/x] = \varphi[\psi'/x]\,. \tag{1.73}$$

For the inductive step, assume $\varphi = c(\varphi_1, \dots, \varphi_n)$ and $\varphi_i[\psi/x] \equiv \varphi_i[\psi'/x]$. For every valuation a, we have

$$a(\varphi[\psi/x]) = a(c(\varphi_1[\psi/x], \dots, \varphi_n[\psi/x])) = c_\Omega(a(\varphi_1[\psi/x]), \dots, a(\varphi_n[\psi/x])) \tag{1.74}$$

$$= c_\Omega(a(\varphi_1[\psi'/x]), \dots, a(\varphi_n[\psi'/x])) = a(c(\varphi_1[\psi'/x], \dots, \varphi_n[\psi'/x]) \tag{1.75}$$

$$= a(\varphi[\psi'/x]) \tag{1.76}$$

The first and last equality follow from the definition of substitution (definition 1.16); the second and fourth from the definition of valuation (formula (1.39)); the third from the inductive hypothesis. Since this holds for every $a \in \Omega^V$, we have $\varphi[\psi/x] \equiv \varphi[\psi'/x]$. ② This is simply associativity of composition of functions. Write $s(\varphi) := \varphi[\psi/x]$ and observe that for every valuation v, $v[\psi/x] := v \circ s$ is a valuation: in fact in diagram (1.77) the left rectangle commutes because s is a substitution and the right rectangle commutes because v is a valuation; therefore, the outer rectangle commutes and this is equivalent to say that $v[\psi/x]$ is a valuation.

$$\begin{tikzcd}
L^n \arrow[rr, dashed, "{v[\psi/x]^n}"] \arrow[dr, "{s^n = [\psi/x]^n}"'] \arrow[ddd, "c"'] & & \Omega^n \arrow[ddd, "c_\Omega"] \\
& L^n \arrow[ur, "v^n"'] \arrow[d, "c"] & \\
& L \arrow[dr, "v"] & \\
L \arrow[ur, "{s = [\psi/x]}"] \arrow[rr, dashed, "{v[\psi/x]}"'] & & \Omega
\end{tikzcd} \tag{1.77}$$

We then have

$$v(\varphi[\psi/x]) = v(s(\varphi)) = v[\psi/x]\,(\varphi) = v[\psi/x]\,(\varphi') = v(s(\varphi')) = v(\varphi'[\psi/x]) \tag{1.78}$$

where the first and last equality follow from the definition of s, the second and the fourth from the definition of $v[\psi/x]$ and the third from the fact that $v[\psi/x]$ is a valuation and $\varphi \equiv \varphi'$. Since this is true for all valuations v, we have $\varphi[\psi/x] \equiv \varphi'[\psi/x]$. ③ The last formula is a consequence of the first two because

$$\varphi[\psi/x] \equiv \varphi[\psi'/x] \equiv \varphi'[\psi'/x] \tag{1.79}$$

where the first equivalence follows from the first formula and the second equivalence from the seconf formula. □

A special case of the lemma is worth mentioning.

Corollary 1.37. (Semantical congruence) Let $x = (x_1, \dots, x_n)$ be a sequence of variables of length n and let $\chi, \psi \in L^n$ be two sequences of formulas of length n. If $\chi \equiv \psi$ are equivalent, then so are the formulas $c(\chi)$ and $c(\psi)$ for every connective c of arity n. In formulas,

$$\chi \equiv \psi \Rightarrow c(\chi) \equiv c(\psi) . \tag{1.80}$$

□

The content of corollary 1.37 can be summarized by saying that semantical equivalence is a congruence on the C-algebra L; we will discuss the consequences of this fact in the chapter on algebraic structures. Here is how the substitution lemma is used.

Proposition 1.38. The following equivalences hold for formulas of L.

$(\varphi \wedge \chi) \wedge \psi \equiv \varphi \wedge (\chi \wedge \psi)$	$(\varphi \vee \chi) \vee \psi \equiv \varphi \vee (\chi \vee \psi)$	associativity	(1.81)
$\varphi \wedge \chi \equiv \chi \wedge \varphi$	$\varphi \vee \chi \equiv \chi \vee \varphi$	commutativity	(1.82)
$\varphi \wedge (\chi \vee \psi) \equiv (\varphi \wedge \chi) \vee (\varphi \wedge \psi)$	$\varphi \vee (\chi \wedge \psi) \equiv (\varphi \vee \chi) \wedge (\varphi \vee \psi)$	distributivity	(1.83)
$\varphi \wedge (\chi \vee \varphi) \equiv \varphi$	$\varphi \vee (\chi \wedge \varphi) \equiv \varphi$	absorption	(1.84)
$\varphi \wedge \top \equiv \varphi$	$\varphi \vee \bot \equiv \varphi$	neutral element	(1.85)
$\varphi \wedge \neg\varphi \equiv \bot$	$\varphi \vee \neg\varphi \equiv \top$	complement	(1.86)

Proof. This is a straightforward application of the substitution lemma. To prove associativity of conjunction, for example, one first proves that $(x \wedge y) \wedge z \equiv x \wedge (y \wedge z)$ using a truth table.

x	y	z	$(x \wedge y) \wedge z$	$x \wedge (y \wedge z)$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

(1.87)

Setting $\sigma := [\varphi/x, \chi/y, \psi/z]$ the substitution lemma now yields

$$(\varphi \wedge \chi) \wedge \psi = [(x \wedge y) \wedge z]\sigma \equiv [x \wedge (y \wedge z)]\sigma = \varphi \wedge (\chi \wedge \psi) . \tag{1.88}$$

The proof for the other formulas is similar and is omitted. □

□ **Uniform notation.** We introduce an alternative to truth tables to determine whether a set of formulas is satisfiable. The method is based on a special notation for formulas.

Definition 1.39. Given a propositional language L, its formulas of *conjunctive type* or *α-formulas* and its formulas of *disjunctive type* or *β-formulas* are defined by the tables below together with their *components* α_i and β_i.

α	α_1	α_2
$\varphi \wedge \chi$	φ	χ
$\neg(\varphi \vee \chi)$	$\neg\varphi$	$\neg\chi$
$\neg(\varphi \to \chi)$	φ	$\neg\chi$

β	β_1	β_2
$\neg(\varphi \wedge \chi)$	$\neg\varphi$	$\neg\chi$
$\varphi \vee \chi$	φ	χ
$\varphi \to \chi$	$\neg\varphi$	χ

(1.89)

The idea behind the definition is to replace α-formulas with the conjunction of their components and β-formulas with their disjunction as follows.

Proposition 1.40. For every α-formula and β-formula in L, we have

$$\alpha \equiv \alpha_1 \wedge \alpha_2, \qquad \beta \equiv \beta_1 \vee \beta_2. \tag{1.90}$$

Proof. It suffices to construct truth tables and use the substitution lemma. We provide details for α-formulas of the third type. From the truth table

x	y	$x \to y$	$\neg(x \to y)$	$\neg y$	$x \wedge \neg y$
0	0	1	0	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	1	1	0	0	0

(1.91)

looking at columns 4 nd 6 we conclude that $\neg(x \to y) \equiv x \wedge \neg y$. Setting $\sigma = [\varphi/x,\ \chi/y]$ and using the substitution lemma we find

$$\neg(\varphi \to \chi) = \neg(x \to y)\sigma \equiv (x \wedge \neg y)\sigma = \varphi \wedge \neg\chi. \tag{1.92}$$

All other equivalences are proved with the same technique. □

Note that the components only involve either the immediate subformulas or their negations. This is the reason for which biimplication does not appear in formula (1.89), because $\varphi \leftrightarrow \psi$ can not be written in this form. When we will use uniform notation we will exclude biimplication from the list of connectives and instead agree that biimplication is defined as the conjunction of two implications,

$$\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi) \tag{1.93}$$

Thus biimplication reduces to a conjunction and therefore to an α-formula.
There are special versions of induction and recursion for uniform notation.

Theorem 1.41. (Structural induction for uniform notation) Suppose L is a propositional language. Every formula of L has a property P if and only if the following conditions are satisfied.

1. If φ is atomic, then $\varphi, \neg\varphi \in P$.
2. If $\varphi \in P$, then $\neg\neg\varphi \in P$.
3. If $\alpha_1, \alpha_2 \in P$, then $\alpha \in P$.
4. If $\beta_1, \beta_2 \in P$, then $\beta \in P$.

Proof. Only sufficiency requires proof. Let $Q = \{\varphi \in L : \varphi, \neg\varphi \in P\}$. We use structural induction for standard notation to prove that $Q = L$.

- If φ is atomic, then $\varphi \in Q$ by condition 1.
- If $\varphi \in Q$, then $\varphi, \neg\varphi \in P$ by definition of Q and therefore $\neg\neg\varphi \in P$ by condition 2. Thus, $\neg\varphi, \neg\neg\varphi \in P$ and therefore $\neg\varphi \in Q$
- Suppose now that $\varphi = \varphi_1 \square \varphi_2$ with $\varphi_i \in Q$. If φ is an α-formula then by (1.89) $\alpha_i = \varphi_i$ because the outermost connective is binary and only the first case in the table can occur; since $\varphi_i \in Q$, $\alpha_i = \varphi_i \in P$ and $\varphi = \alpha \in P$ by condition 3; also $\neg\varphi$ is a β-formula,necessarily of the first type, with $\beta_i = \neg\varphi_i \in P$ and therefore $\neg\varphi = \beta \in P$ by condition 4; since $\varphi, \neg\varphi \in P$, we have $\varphi \in Q$.
  If φ is a β-formula, then it must be one of the last two types, and therefore β_1 is either φ_1 or $\neg\varphi_1$ and $\beta_2 = \varphi_2$. Either way, since $\varphi_i, \neg\varphi_i \in P$, we have that $\beta_1, \beta_2 \in P$ and $\varphi = \beta \in P$ by condition 4; also, $\neg\varphi$ is an α-formula of the last two types, hence α_1 is either $\neg\varphi_1$ or φ_1 and $\alpha_2 = \neg\varphi_2$; either way, $\alpha_1, \alpha_2 \in P$ and therefore $\neg\varphi = \alpha \in P$. Thus $\varphi \in Q$ also in this case.

By theorem 1.8, $Q = L$. Thus $L = Q \subseteq P \subseteq L$ and therefore $P = L$. □

In a similar fashion, from the structural recursion theorem one obtains the following

Theorem 1.42. (Structural recursion for uniform notation) Assume L is a propositional language with variables in X and $f_A : A \to S$ is a function from the set of atomic formulas and their negations to an arbitrary set. Assume further that we are given functions

$$f_{\neg\neg} : S \to S, \qquad f_\alpha, f_\beta : S^2 \to S. \tag{1.94}$$

Then f_A has a unique extension to a function $f : L \to S$ such that

- $f(\neg\neg\varphi) = f_{\neg\neg}(\varphi)$ for every formula φ;
- $f(\alpha) = f_\alpha(\alpha_1, \alpha_2)$ for every conjunctive formula α;
- $f(\beta) = f_\beta(\beta_1, \beta_2)$ for every disjunctive formula β.

Example 1.43 (Rank) The *rank* of formulas is the function $r : L \to \mathbf{N}$ defined as follows. For atomic formulas and their negations $r_A(\varphi) = 0$ except for $r_A(\neg\bot) = r_A(\neg\top) = 1$. Also, $r_{\neg\neg} : \mathbf{N} \to \mathbf{N}$ is defined by $r_{\neg\neg}(n) = n + 1$ and $r_\alpha = r_\beta : \mathbf{N}^2 \to \mathbf{N}$ are defined as $r_\alpha(m, n) = m + n + 1$. Then by the uniform version of the definition by structural recursion theorem, r_A extend uniquely to a function $r : L \to \mathbf{N}$ compatible with the assigned data.
For a concrete example, let $\varphi = (\top \to x) \to y$.

$$r(\varphi) = r(\neg(\top \to x)) + r(y) + 1 = [r(\top) + r(\neg x) + 1] + 0 + 1 \tag{1.95}$$
$$= [0 + 0 + 1] + 1 = 2 \tag{1.96}$$

The rank function counts the nontrivial nodes in a tableau expansion.

Example 1.44 (Depth) The *depth* of formulas is the function $h : L \to \mathbf{N}$ recursively defined by the following conditions:

1. If x is a variable, then $h(x) = h(\neg x) = 0$
2. $h(\top) = h(\bot) = 0$
3. $h(\neg\top) = h(\neg\bot) = 1$
4. $h(\neg\neg\varphi) = h(\varphi) + 1$
5. $h(\alpha) = \max(h(\alpha_i)) + 1$
6. $h(\beta) = \max(h(\beta_i)) + 1$

The depth of a formula is a measure of the number of complexity with respect to semantical equivalence.

□ **Consequence.** This is the central comcept in semantics, as it embodies the classical notion of theorem in mathematics

DEFINITION 1.45. A formula φ is a *consequence* of a set of formulas $F \subseteq L$ if all models of F are also models of φ. In this case we write $F \vDash \varphi$ and also say that F *satisfies* φ. We write $\vDash \varphi$ when F is the empty set.

F is the set of *premises* and φ is the *conclusion*. The idea behind consequence is that if $F \vDash \varphi$ and all the premises are satisfied, then so is the conclusion, which is exactly the way theorems in mathematics are formulated. The notation for consequence can be simplified omitting the parenhese around F, so that one writes $\varphi_1, \varphi_2, \ldots \vDash \varphi$ instead of $\{\varphi_1, \varphi_2, \ldots\} \vDash \varphi$. We write $F \nvDash \varphi$ to indicate that φ is not a consequence of F; what this means is that there is at least a valuation a such that $a \vDash F$ but $a \nvDash \varphi$. Some special cases deserve consideration.

- $\vDash \varphi$ means that φ is valid. In fact, since every valuation satisfies the empty set, $\varnothing \vDash \varphi$ means that every valuation satisfies φ.
- $F \vDash \bot$ means that F is unsatisfiable. This is because if $a \vDash F$ then we should have $a \vDash \bot$, which is impossible.

Before discussing example, we provide an alternative formulation of consequence in terms of satisfiability.

PROPOSITION 1.46. Given $F \subseteq L$, a formula $\varphi \in L$ is a consequence of F if and only if the set $F \cup \{\varphi\}$ is unsatisfiable. In symbols,

$$F \vDash \varphi \;\Leftrightarrow\; F \cup \{\neg\varphi\} \vDash \bot. \tag{1.97}$$

Proof. Since the statements we are discussing can only be either true or false, in order to prove that the left hand side of (1.97) is true if and only if the right hand side is, we will instead prove that the left hand side is false if and only if the right hand side is also false, as this is easier to show. The claim now follows from the chain of equivalent assertions

$$\begin{aligned} F \nvDash \varphi &\Leftrightarrow \text{there exists } a \in \Omega^V \text{ such that } a \vDash F,\ \ a \nvDash \varphi && (1.98)\\ &\Leftrightarrow \text{there exists } a \in \Omega^V \text{ such that } a \vDash F,\ \ a \vDash \neg\varphi && (1.99)\\ &\Leftrightarrow \text{there exists } a \in \Omega^V \text{ such that } a \vDash F \cup \{\neg\varphi\} && (1.100)\\ &\Leftrightarrow F \cup \{\neg\varphi\} \nvDash \bot && (1.101) \end{aligned}$$

in which the first claim follows from the definition of consequence, the second from the truth table of negation, the third from the definition of consequence and similarly for the last one, with, in addition, the fact that no valuation satisfies $\bot$. □

Example 1.47 In the propositional language L generated by the variables $\{x, y, z\}$ we have

$$x \to y,\quad y \to z \vDash x \to z. \tag{1.102}$$

The most elementary way of proving this is to compute the truth table for all formulas, as shown oin (1.103) on the left. From the table it is apparent that the models of the set of premises F are the valuations $\{v_1, v_2, v_4, v_8\}$, the index corresponding to the rows of the table. The valuations are also models of the conclusion φ — even though φ has additional models. Therefore, φ is a consequence of F.

x	y	z	$x \to y$	$y \to z$	$x \to z$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	1

(1.103)

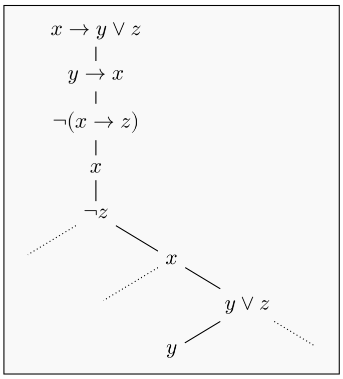
As an alternative, one can use proposition 1.46 and prove that $F \cup \{\varphi\}$ is unsatisfiable; this can be proved by showing that $F \cup \{\varphi\}$ has a closed semantic tableau and one such tableau is given in (1.102) on the right.

Example 1.48 In the propositional language L generated by $V = \{x, y, z\}$ and the standard set of connectives,

$$x \to y \vee z,\ y \to x \not\models x \to z. \tag{1.104}$$

To see this, consider first the truth tables of the 3 formulas, shown below. Observe that valuation v_7 in the seventh row satisfies the set of premises F but not the conclusion φ and therefore φ is not a consequence of F.

x	y	z	$x \to y \vee z$	$y \to x$	$x \to z$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	1



(1.105)

To achieve the same result with a semantic tableau we have to prove that $F \cup \{\neg\varphi\}$ is satisfiable, i.e. that it has a tableau with an open branch. One such tableau is shown above.

We investigate the relation between consequence and connectives.

Proposition 1.49. If $F \subseteq L$, φ is any formula and α, β are respectively formulas of conjunctive or disjunctive type, then the following assertions hold.

$$F \models \neg\varphi \quad \Leftrightarrow \quad F \cup \{\varphi\} \models \bot \tag{1.106}$$

$$F \models \alpha \quad \Leftrightarrow \quad F \models \alpha_1 \text{ and } F \models \alpha_2 \tag{1.107}$$

$$F \models \beta \quad \Leftrightarrow \quad F \cup \{\neg\beta_1\} \models \beta_2 \tag{1.108}$$

Proof. ① By proposition 1.46 $F \vDash \neg\varphi$ if and only if $F \cup \{\neg\neg\varphi\}$ is unsatisfiable; since $\neg\neg\varphi \equiv \varphi$, this happens if and only if $F \cup \{\varphi\}$ is unsatisfiable. ② Since $\alpha \equiv \alpha_1 \wedge \alpha_2$, we have the following chain of equivalent statements.

$$\begin{aligned} F \vDash \alpha &\Leftrightarrow \forall a \in \Omega^V \ (a \vDash F \Rightarrow a \vDash \alpha) && (1.109)\\ &\Leftrightarrow \forall a \in \Omega^V \ (a \vDash F \Rightarrow a \vDash \alpha_1 \text{ and } a \vDash \alpha_2) && (1.110)\\ &\Leftrightarrow \forall a \in \Omega^V \ ((a \vDash F \Rightarrow a \vDash \alpha_1) \text{ and } (a \vDash F \Rightarrow a \vDash \alpha_2)) && (1.111)\\ &\Leftrightarrow \forall a \in \Omega^V \ (a \vDash F \Rightarrow a \vDash \alpha_1) \text{ and } \forall a \in \Omega^V \ (a \vDash F \Rightarrow a \vDash \alpha_2)) && (1.112)\\ &\Leftrightarrow F \vDash \alpha_1 \text{ and } F \vDash \alpha_2. && (1.113) \end{aligned}$$

③ It is more convenient to show that the the two statements are false in the same cases. Recalling that $\beta \equiv \beta_1 \vee \beta_2$ and that a disjunction is false only when both disjuncts are, we have:

$$\begin{aligned} F \nvDash \beta &\Leftrightarrow \exists a \in \Omega^V \ (a \vDash F \text{ and } a \nvDash \beta) && (1.114)\\ &\Leftrightarrow \exists a \in \Omega^V \ (a \vDash F \text{ and } a \nvDash \beta_1 \text{ and } a \nvDash \beta_2) && (1.115)\\ &\Leftrightarrow \exists a \in \Omega^V \ (a \vDash F \text{ and } a \vDash \neg\beta_1 \text{ and } a \nvDash \beta_2) && (1.116)\\ &\Leftrightarrow \exists a \in \Omega^V \ (a \vDash F \cup \{\neg\beta_1\} \text{ and } a \nvDash \beta_2) && (1.117)\\ &\Leftrightarrow F \cup \{\neg\beta_1\} \nvDash \beta_2. && (1.118) \end{aligned}$$

A special case is worth mentioning explicitly, as its syntactivcal version will play an important role in Hilbert calculus: if $\beta = \varphi \to \psi$ then $\neg\beta_1 = \neg\neg\varphi \equiv \varphi$ and formula (1.108) immediately yields

Corollary 1.50. (Deduction theorem, semantical version) If $F \subseteq L$ and $\varphi, \psi \in L$ are arbitrary formulas, then

$$F \vDash \varphi \to \psi \quad \Leftrightarrow \quad F \cup \varphi \vDash \psi \tag{1.119}$$

□

Example 1.51 It is important to observe that in the case of formulas of disjunctive type

$$F \vDash \beta \not\Leftrightarrow F \vDash \beta_1 \text{ or } F \vDash \beta_2. \tag{1.120}$$

In trying to reproduce the proof for the conjunctive case one realizes that there are two places in which the proof fails. First

$$(\varphi \to \alpha) \equiv (\varphi \to \alpha_1) \wedge (\varphi \to \alpha_2), \qquad (\varphi \to \beta) \not\equiv (\varphi \to \beta_1) \vee (\varphi \to \beta_2). \tag{1.121}$$

The formula on the left has been implicitly used in (1.111) and in the formula on the right only the implication right to left holds. Second, in (1.112) we have used the fact that asserting that every $a \in \Omega^V$ has both properties P and Q is equivalent to say that every a has property P and every a has property Q. By contrast, asserting that every a has either property P or Q does not imply that every a has property P or that every a has property Q as for some a's property P might hold with Q holding in the remaining cases.
For a concrete example, let $\beta = x \vee y$ and $F = \{x \vee y\}$. Then $x \vee y \vDash x \vee y$, so that $F \vDash \beta$. However $x \vee y \nvDash x$, as is seen by considering the valuation defined by $a(x) = 0$ and $a(y) = 1$ and likewise $x \vee y \nvDash y$ for $a(x) = 1$ and $a(y) = 0$, so that neither $F \vDash \beta_1$ nor $F \vDash \beta_2$ hold.
Notice, however, that $F \vDash \beta_i \Rightarrow F \vDash \beta$ because $\beta_i \vDash \beta$. It is the reverse impication that fails.

Example 1.52 Along the same lines, it is worth observing that

$$F \vDash \neg\varphi \not\Leftarrow F \nvDash \varphi. \tag{1.122}$$

In fact the implication left to right is correct:

$$\begin{aligned} F \vDash \neg\varphi &\Rightarrow \forall a \in \Omega^V (a \vDash F \Rightarrow a \vDash \neg\varphi) && (1.123)\\ &\Rightarrow \forall a \in \Omega^V (a \vDash F \Rightarrow a \nvDash \varphi) && (1.124)\\ &\Rightarrow F \nvDash \varphi. && (1.125) \end{aligned}$$

The reverse implication, though, is false in general, the reason being that neithr φ nor $\neg\varphi$ need to be consequences of F. For example, if we take $F = \{x \vee y\}$ and $\varphi = x$, then $F \nvDash \varphi$ (take $a(x) = 0$ and $a(y) = 1$) but also $F \nvDash \neg\varphi$ (take any valuation with $a(x) = 1$).

Observe that the consequence symbol is not part of the object language and therefore the assertion $F \vDash \varphi$ is not a formula of propositional logic, but rather a statement in the metalanguage. We can, however, internalize this statement as a formula of propositional logic, at least when F is finite, in the sense that $F \vDash \varphi$ is true exactly when the corresponding formula is valid.

Corollary 1.53. (Internalization of consequence) For any finite $F = \{\varphi_1, \dots, \varphi_n\} \subseteq L$, we have that $F \vDash \psi$ if and only $\varphi \to \psi$ is valid, φ being the conjunction of the formulas in F. In symbols,

$$\varphi_1, \dots, \varphi_n \vDash \psi \Leftrightarrow \vDash (\varphi_1 \wedge \dots \wedge \varphi_n) \to \psi. \tag{1.126}$$

Proof. Setting $\varphi := \bigwedge_{i=1}^{n} \varphi_i$, we have a chain of equivalent claims

$$\varphi_1, \dots, \varphi_n \vDash \psi \quad \Leftrightarrow \quad \varphi \vDash \psi \quad \Leftrightarrow \quad \vDash \varphi \to \psi$$

where the first equivalence follows from the fact that for every valuation $a \in \Omega^V$ we have that $a \vDash F$ when a satisfies all the φ_i's which, by definition of conjunction, happens exactly when $a \vDash \varphi$. The second equivalence is an instance of the deduction theorem. □

As a final observation it is worth mentioning that $F \vDash \varphi$ precisely when every truth set containing F also cantains φ. Since truth sets do not need reference to Ω, it turns out that one can characterize consequence without using semantics directly.

□ **Functional completeness and normal forms.** An application of equivalence is the possibility to reduce the number of connectives.

Definition 1.54. A set C of connectives is *functionally complete* if every formula $\varphi \in L$ is equivalent to a formula containing only connectives from C.

Complete sets of connectives are used to reduce the number of connectives and therefore simplifying proofs that use structural induction. On the negative side, it becomes more difficult to express concepts in a language with fewer connectives.

Proposition 1.55. Each of the following sets of connectives is complete.

$$\{\bot, \to\}, \qquad \{\top, \neg, \wedge\}, \qquad \{\bot, \neg, \vee\}. \tag{1.127}$$

Proof. If C is a set of connectives, let P be the set of formulas containing only connectives from C and Q the set of formulas equivalent to one in P. P is closed under the connective of C and Q is closed under equivalence and since it contains P, it is also closed under the connectives of C by the substitution lemma. We will prove that $Q = L$ by uniform structural induction.
① If $C = \{\bot, \to\}$ and $\varphi \in Q$ then $\neg\varphi \equiv \varphi \to \bot \in Q$ and Q is also closed under negation. We now use induction.

- ○ Suppose φ is atomic. All variables and $\bot$ are in P and hence in Q and all atomic formulas and their negations are either of this type or the negation of formulas of this type; since Q is closed under negation, both φ and $\neg\varphi$ are in Q.
- ○ If $\varphi \in Q$ then $\neg\neg\varphi \equiv \varphi$ hence $\neg\neg\varphi \in Q$.
- ○ If $\varphi = \alpha$ and $\alpha_1, \alpha_2 \in Q$, then $\alpha \equiv \alpha_1 \wedge \alpha_2 \equiv \neg(\alpha_1 \to \neg\alpha_2)$ and $\alpha \in Q$.
- ○ If $\varphi = \beta$ and $\beta_1, \beta_2 \in Q$, then $\beta \equiv \beta_1 \vee \beta_2 \equiv \neg\beta_1 \to \beta_2$ and therefore $\beta \in Q$.

② If $C := \{\top, \neg, \wedge\}$, we use induction as follows.

- ○ Assume φ is atomic. If φ is either a variable or $\top$ then $\varphi, \neg\varphi \in P \subseteq Q$. If $\varphi = \bot$ then $\varphi \equiv \neg\top \in P$ and $\neg\varphi \equiv \top \in P$ and again $\varphi, \neg\varphi \in Q$.
- ○ If $\varphi \in Q$ then $\neg\neg\varphi \equiv \varphi$ hence $\neg\neg\varphi \in Q$.
- ○ If $\varphi = \alpha$ and $\alpha_1, \alpha_2 \in Q$, then $\alpha \equiv \alpha_1 \wedge \alpha_2 \in Q$
- ○ If $\varphi = \beta$ and $\beta_1, \beta_2 \in Q$, then $\beta \equiv \beta_1 \vee \beta_2 \equiv \neg(\neg\beta_1 \wedge \neg\beta_2) \in Q$

The third case is similar to the second. □

It is worth observing that if L has at least one variable x, then

$$\top \equiv \neg(x \wedge \neg x), \qquad \bot \equiv \neg(x \vee \neg x) \tag{1.128}$$

so that $\{\top, \neg, \wedge\}$ and $\{\top, \neg, \wedge\}$ in (1.127) can be replaced respectively by $\{\neg, \wedge\}$ and $\{\neg, \vee\}$. Not only are these sets of connectives complete; they also suffice to describe any possible connective one could imagine, as we now show.

Proposition 1.56. If L has a finite set of variables V, then every function $\omega : \Omega^V \to \Omega$ is the interpretation of some formula $\varphi \in L$.

Proof. Observe that if $|V| = n$ is finite, then $|\Omega^V| = 2^n$ is also finite. If $S = \{a \in \Omega^V : \omega(a) = 1\}$ is the set of valuations that satisfy ω, for every $a \in S$ define

$$\hat{a}(x_i) = \begin{cases} x_i & if \text{a(x\_i)=1} \\ \neg x_i & if \text{a(x\_i)=0} \end{cases} \qquad \varphi_a = \bigwedge_{x_i \in V} \hat{x}_i. \tag{1.129}$$

In other words, φ_a contains the conjunctions of variables on which a is true and of the negation of those on which a is false. Thus, $\varphi_a(a) = 1$ and $\varphi_a(b) = 0$ for all other valuations $b \neq a$. Thus, if we define

$$\varphi = \bigvee_{a \in S} \varphi_a \tag{1.130}$$

then $\varphi \in L$ is a formula which is satisfied exactly by the valuations in S and therefore $[\![\varphi]\!] = \omega$. □

Example 1.57 In the propositional language L generated by the variables $\{x, y, z\}$ and the standard set of connectives, consider the truth function ω described by the table below.

x	y	z	ω
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

(1.131)

The set of valuations satisfying ω is $S = \{a_1, a_2, a_6\}$ corresponding to rows 1, 2 and 6 of the table. The corresponding formulas are

$$\varphi_1 = \neg x \wedge \neg y \wedge \neg z, \qquad \varphi_2 = \neg x \wedge \neg y \wedge z, \qquad \varphi_6 = x \wedge \neg y \wedge z \tag{1.132}$$

and a formula with the same interpretation of ω is $\varphi = \varphi_1 \vee \varphi_2 \vee \varphi_6$.

The idea used in the previous proof of writing a formula as a disjunction of conjunctions of variables and their negations has important generalizations.

Definition 1.58. A *literal* is a formula which is either a propositional variable x, or the negation of a propositional variable $\neg x$ or a constant, $\top$ or $\bot$. A *clause* is a finite disjunction of literals and a *dual clause* is a finite conjuction of literals.

A clause is *reduced* if

- no literal is repeated,
- no literal occurs with opposite signs,
- $\bot$ is omitted when at least one other literal is present.

Observe that every clause is equivalent to one in reduced form: since $\varphi \vee \varphi \equiv \varphi$ by idempotency of disjunction, we can omit repetitions of literals; since $\varphi \vee \neg\varphi \equiv \top$ which is abosrbing for disjunction, every clause containing both φ and $\neg\varphi$ is equivalent to $\top$; since $\varphi \vee \bot \equiv \varphi$ by the neutral element formula, $\bot$ can be omitted if another literal is present. Also, because of the last condition we will say that the clause reduced to $\bot$ is the empty clause. Similarly, a dual clause is reduced if no literals are repeated, no literal occurs with opposite signs and $\top$ is omitted. We will identify the empty dual clause with $\top$.

Definition 1.59. A formula φ is in *conjunctive normal form* (CNF) if it is a conjunction of clauses and in *disjunctive normal form* (DNF) if it is finite disjunction of dual clauses. We say that φ admits a conjunctive normal form if it is equivalent to a formula in conjunctive normal form and that it admits a disjunctive normal form if it is equivalent to a formula in disjunctive normal form.

Observe that if a formula admits a conjunctive or a disjunctive normal form, this is by no means unique. However, existence of either implies the existence of both normal forms.

Lemma 1.60. A formula $\varphi \in L$ admits a conjunctive normal form if and only if it admits a disjunctive normal form.

Proof. Assume that φ_i is a clause with literals λ_{ij}. Note that we can assume that $1 \leq j \leq n$ with a single n for every i, possibly adding literals of type $\lambda_{ij} = \bot$. Using distributivity of conjunction over disjunction we find

$$\bigwedge_{i=1}^{m} \varphi_i \equiv \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n} \lambda_{ij} \right) \equiv (\lambda_{11} \vee \ldots \vee \lambda_{1n}) \wedge \ldots \wedge (\lambda_{m1} \vee \ldots \vee \lambda_{mn}) \tag{1.133}$$

$$\equiv \bigvee_{j_1, \ldots, j_m = 1}^{n} (\lambda_{1j_1} \wedge \ldots \wedge \lambda_{mj_m}) \equiv \bigvee_{j_1, \ldots, j_m = 1}^{n} \left(\bigwedge_{i=1}^{m} \lambda_{ij_i} \right) \equiv \bigvee_{j_1, \ldots, j_m = 1}^{n} \psi_{j_1, \ldots, j_m} \tag{1.134}$$

and the last formula is a disjunction of dual clauses. Thus, every formula admitting a CNF also admits a DNF. If we exchange conjunction and disjunction in the proof above and use distributivity of disjunction over conjunction we obtain that every formula admitting a DNF also has a CNF. □

Observe that by duality the negation of a formula in CNF is a formula in DNF and conversely. Thus, in the proof of the lemma, once we know that every formula has a CNF we could argue that $\neg\varphi$ in particular has one and taking negations we obtain that φ has a DNF.

Proposition 1.61. Every formula $\varphi \in L$ admits a conjunctive and a disjunctive normal form.

Proof. Let $P \subseteq L$ be the set of formulas admitting both a conjunctive and a disjunctive normal form. P is closed under equivalence, because a formula equivalent to one admitting a normal form also admits the same normal form. Also, P is closed under negation because the negation of a formula in CNF is a formula in DNF and conversely. Also P is closed under conjunction, because the conjunction of two formulas in CNF is a formula in CNF, because of lemma 1.60 and because P is closed under equivalence. Similarly, P is closed under dijunctions because the disjunction of two formulas in DNF is also in DNF. We prove by structural in uniform notation that $P = L$.

- If φ is atomic then it is a literal and therefore it is both in CNF and DNF. Since P is closed under negation, also $\neg\varphi \in P$.
- If $\varphi \in P$, then $\neg\neg\varphi \equiv \varphi \in P$.
- If $\varphi = \alpha$ is an α-formula and $\alpha_1, \alpha_2 \in P$, then $\varphi \equiv \alpha_1 \wedge \alpha_2 \in P$.
- If $\varphi = \beta$ is a β-formula and $\beta_1, \beta_2 \in P$, then $\varphi \equiv \beta_1 \vee \beta_2 \in P$.

Thus $P = L$ and every formula has both CNF and DNF. □

Although proposition 1.61 guarantees the existence of normal forms, it does not provide any means of computing them effectively. We will therefore describe an algorithm that produces a conjuctive normal form for any given formula. First a *generalized clause* is a disjunction of formulas which we will refer to as the *components* of the generalid clause. we will consider formulas which are conjunctions of generalized clauses as in

$$\varphi = \bigwedge_{i=1}^{m} \varphi_i = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n} \varphi_{ij} \right) \tag{1.135}$$

where each φ_i is a generalized clause of φ and $\{\varphi_{i1}, \ldots, \varphi_{in}\}$ is the set of components of φ_i. Notice that we have implicitly assumed for simplicity that each generalized clause has the same number of components, which is not an issue from the semantical viewpoint, as we can always add components of the form $\bot$ which are neutral for disjunction. Observe that any formula can be written in this form as $\varphi = \varphi_1$ with a single generalized clause in which the only component is φ itself. Also observe that there may be more than one way of writing a formula in this form. The algorithm works as follows.

1. Start by writing the formula as a single generalized clause with a single component.
2. Replace all occurrences of $\neg\bot$ with $\top$ and all occurrences of $\neg\top$ with $\bot$.
3. Replace components of the form $\neg\neg\varphi_{ij}$ with φ_{ij}.

4. If a component of a generalized clause φ_i is an α-formula, replace φ_i with the two generalized clauses $\varphi_i[\alpha_1/\alpha]$ and $\varphi_i[\alpha_2/\alpha]$, in which α is replaced first by α_1 and then by α_2.
5. If a β-formula, occurs in a generalized clause φ_i replace φ_i with $\varphi_i[\beta_1 \vee \beta_2/\beta]$.

The algorithm stops when no component can be further expanded, that is when the components are literals. This means that the original formula has been replaced by one in conjunctive normal form and since every step of the algorithm produces an equivalent formula, the final expansion is a CNF of the original formula. Observe, however, that the algorithm as stated is not deterministic, as we may have the option at any step to expand different generalized clauses and we could choose any of them. Dualizing the rules for α and β-formulas above one obtains an algorithm for disjunctive normal form.

Example 1.62 We compute a CNF for the formula $(x \to (y \to z)) \to (x \wedge y \to z)$.

Step	Formula	Rule
1	$(x \to (y \to z)) \to (y \wedge z \to x)$	
2	$\neg(x \to (y \to z)) \vee (y \wedge z \to x)$	β-expansion
3	$\neg(x \to (y \to z)) \vee \neg(y \wedge z) \vee x$	β-expansion
4	$\neg(x \to (y \to z)) \vee \neg y \vee \neg z \vee x$	β-expansion
5	$(x \vee \neg y \vee \neg z \vee x) \wedge (\neg(y \to z) \vee \neg y \vee \neg z \vee x)$	α-expansion
6	$(x \vee \neg y \vee \neg z \vee x) \wedge (y \vee \neg y \vee \neg z \vee x) \vee \wedge(\neg z \vee \neg y \vee \neg z \vee x)$	α-expansion

(1.136)

Line 6 contains a conjunctive normal form of φ as produced by the algorithm. Note, however, that this can be further simplified: the second clause contains both y and $\neg y$ and therefore simplifies to $\top$ which can be omitted being neutral for conjunction; the two remaining clases are equivalent so that φ has a CNF reduced to the single clause $(x \vee \neg y \vee \neg z)$.

1.4 Calculi

In this section we examine abstractly the structure of calculi and the conditions they have to satisfy to be compatible with semantics.

☐ **Inference relations.** Fix a propositional language L.

Definition 1.63. A relation $\vdash : PR \rightharpoonup L$ is an *inference relation* if it satisfies the following conditions.

$$F \vdash \varphi \iff E \vdash \varphi \text{ for some finite subset } E \subseteq F \qquad \text{compactness} \qquad (1.137)$$

$$\varphi \in F \Rightarrow F \vdash \varphi \qquad \text{assumption} \qquad (1.138)$$

$$F \vdash \varphi \text{ and } F, \varphi \vdash \psi \Rightarrow F \vdash \psi \qquad \text{cut} \qquad (1.139)$$

$$F \vdash \varphi \iff F, \neg\varphi \vdash \bot \qquad \text{consistency} \qquad (1.140)$$

We write $F \vdash \varphi$ when the pair (F, φ) belongs to the inference relation and say that φ can be *inferred* from F. When F is empty we write $\vdash \varphi$ and call φ a theorem for the inference relation.

Example 1.64 Semantical consequence is an inference relation. All the axioms are straightforward except for compactness. In view of consistency, this axiom amounts to the fact that $F \vDash \bot$ if and only if there exists a finite subset $E \subseteq F$ such that $E \vDash \bot$; in other words, a set is unsatisfiable if and only if it has a finite unsatisfiable set. Equivalently, this amounts to say that a set F is satisfiable if and only if every finite subset is and this is the content of the compactness theorem that will be proved later on.

A useful consequence of the axioms is the following property.

- Weakening. If $F \vdash \varphi$ and $F \subseteq G$, then $G \vdash \varphi$. This follows from compactness: if $F \vdash \varphi$ then there exists a finite subset $E \subseteq F$ such that $E \vdash \varphi$; however, E is also a finite subset of G and therefore $G \vdash \varphi$.

Our goal here is in fact to construct a syntactically defined inference relation that coincides with semantical consequence. Let us be more precise.

Definition 1.65. An inference relation is

- *sound* if $F \vdash \varphi \Rightarrow F \vDash \varphi$,
- *complete* if $F \vDash \varphi \Rightarrow F \vdash \varphi$.

Thus, a sound and complete inference relation coincides with semantical inference. We can be more precise by saying that our goal is to define by syntactical means sound and complete inference relations. This will provide alternative ways of proving that a formula φ is a consequence of a set $F \subseteq L$. Proving that a particular inference relation is souind is not ususally difficult, as we will see later on. Proving completeness, however, is not that straighforward. For this reason, we introduce a concept that provide an alternative description of completeness.

Definition 1.66. A set of formulas $F \subseteq L$ is *consistent* for an inference relation if $F \nvdash \bot$ and *inconsistent* if $F \vdash \bot$.

When the inference relation is semantical consequence, a set is consistent precisely when it is satisfiable and inconsistent when it is unsatisfiable. In other words, consistency is, at least for the moment, a generalization of satisfiability. We use consistency to provide an alternative characterization of soundness and completeness.

Theorem 1.67. Assume $\vdash$ is an inference relation on L.

- The inference relation is sound if and only if every satisfiable set is consistent.
- The inference relation is complete if and only if every consistent set is satisfiable.

Therefore, $\vdash$ is sound and complete if and only if satisfiable and consistent sets coincide.

Proof. Since $F \vDash \varphi \Leftrightarrow F, \neg\varphi \vDash \bot$ (proposition 1.46), we can use the consistency property of of the inference relation to rephrase soundness and completeness using the conditions

$$F, \neg\varphi \vdash \bot \Rightarrow F, \neg\varphi \vDash \bot, \qquad F, \neg\varphi \vDash \bot \Rightarrow F, \neg\varphi \vdash \bot. \tag{1.141}$$

If every satisfiable set is consistent, then every inconsistent set is unsatisfiable, the first condition in (1.141) is satisfied and the inference relation is sound. Conversely, if the relation is sound then $F \vdash \bot \Rightarrow F \vDash \bot$ which shows that every inconsistent set is unsatisfiable and therefore every satisfiable set must be consistent. If every consistent set is satisfiable, then every unsatisfiable set is inconsistent; the second condition in formula (1.141) is therefore satisfied and the relation is complete. Conversely, if the inference relation is complete then $F \vDash \bot \Rightarrow F \vdash \bot$, showing that every unsatisfiable set is inconsistent and therefore every consistent set is satisfiable. □

Observe that the proof actually uses only the consistency property of an inference relation. In view of the importance of the concept of consistent and inconsistent sets, we collect their basic properties.

Proposition 1.68. In an inference relation, a set $F \subseteq L$ is inconsistent if and only if it has a finite inconsistent subset and F is consistent if and only if all its finite subsets are consistent. Moreover:

$$(F, \varphi \vdash \bot) \wedge (F, \neg\varphi \vdash \bot) \Rightarrow (F \vdash \bot) \tag{1.142}$$

$$F \vdash \varphi \text{ and } F \vdash \neg\varphi \Rightarrow F \vdash \bot \tag{1.143}$$

$$\varphi, \neg\varphi \in F \Rightarrow F \vdash \bot \tag{1.144}$$

Also, formula (1.142) is equivalent to the cut rule.

Proof. The first claim follows from compactness, because $F \vdash \bot$ if and only if there exists a finite subset $E \subseteq F$ such that $E \vdash \bot$ and the second claim is the counterpositive assertion. ① and ② are proved in the two deductions below.

STEP	CLAIM	REASON
1	$F, \varphi \vdash \bot$	hypothesis
2	$F, \neg\varphi \vdash \bot$	hypothesis
3	$F \vdash \varphi$	2, consistency
4	$F \vdash \bot$	3, 1, cut

STEP	CLAIM	REASON
1	$F \vdash \varphi$	hypothesis
2	$F \vdash \neg\varphi$	hypothesis
3	$F, \neg\varphi \vdash \bot$	1, consistency
4	$F, \vdash \bot$	2, 3, cut

(1.145)

③ Ff $\varphi, \neg\varphi \in F$ then $F \vdash \varphi$ and $F \vdash \neg\varphi$ by assumption and therefore $F \vdash \bot$ by 2. The proof that property 1 implies cut with the indicated hypothesys is given below — recall that weakening is a consequence of compactness.

STEP	CLAIM	REASON
1	$F \vdash \varphi$	hypothesis
2	$F, \varphi \vdash \psi$	hypothesis
3	$F, \neg\varphi \vdash \bot$	1, consistency
4	$F, \neg\varphi, \neg\psi \vdash \bot$	3, weakening
5	$F, \varphi, \neg\psi \vdash \bot$	2, consistency
6	$F, \neg\psi \vdash \bot$	4, 5, property 1
7	$F \vdash \psi$	6, consistency

(1.146)

The cut rule and the inconsistency rules (1.142) are equivalent to two seemingly more general formulas, which are useful in proofs.

Proposition 1.69. The cut (1.139) and inconsistency (1.142) rules are respectively equivalent to the formulas

$$F \vdash \varphi \text{ and } G, \varphi \vdash \psi \;\Rightarrow\; F, G \vdash \psi \tag{1.147}$$

$$F, \varphi \vdash \bot \text{ and } G, \neg\varphi \vdash \bot \;\Rightarrow\; F, G \vdash \bot \tag{1.148}$$

Proof. ① It is clear that formula (1.139) is a special case of (1.147) with $F = G$. Conversely, if the cut rule (1.139) holds, we can prove (1.147) as shown on the left below.

STEP	CLAIM	REASON
1	$F \vdash \varphi$	hypothesis
2	$G, \varphi \vdash \psi$	hypothesis
3	$F, G \vdash \varphi$	1, weakening
4	$F, G, \varphi \vdash \psi$	2, weakening
5	$F, G \vdash \psi$	3, 4, cut

STEP	CLAIM	REASON
1	$F, \varphi \vdash \bot$	hypothesis
2	$G, \neg\varphi \vdash \bot$	hypothesis
3	$F, G, \varphi \vdash \bot$	1, weakening
4	$F, G, \neg\varphi \vdash \bot$	2, weakening
5	$F, G \vdash \bot$	3, 4, inconsistency

(1.149)

② Again, formula (1.142) is a special case of (1.148) with $F = G$. Conversely, if (1.142) holds, we can prove (1.148) as sonw on the right above. □

We will refere to the generalized version of the formulas still as to cut and inconsistency.

Proposition 1.70. The following properties hold for any inference.

$$F, \neg\neg\varphi \vdash \varphi \tag{1.150}$$

Proof. ① The double negation rule follows from the following table.

STEP	CLAIM	REASON
1	$F, \neg\varphi \vdash \neg\varphi$	assumption
2	$F, \neg\varphi, \neg\neg\varphi \vdash \bot$	1, consistency
3	$F, \neg\neg\varphi \vdash \varphi$	2, consistency

(1.151)

As we already remarked, proving soundness for an inference relation is not usually difficult. We can prove completeness using theorem 1.67 by showing that every consistent set is satisfiable. We will isolate some properties of the inference relation that imply this property.

□ **Complete regularity.** We isolate a first set set of conditions implying that every consistent set is satisfiable. This set of conditions is based on the classical notion of complete consistent set.

Definition 1.71. An inference relation is *completely regular* if it satisfies the following conditions.

$$F \vdash \alpha \;\Leftrightarrow\; F \vdash \alpha_1 \text{ and } F \vdash \alpha_2 \qquad \alpha\text{-rule} \tag{1.152}$$

$$F, \beta \vdash \bot \;\Leftrightarrow\; F, \beta_1 \vdash \bot \text{ and } F, \beta_2 \vdash \bot \qquad \beta\text{-rule} \tag{1.153}$$

$$F \vdash \top \qquad \text{truth} \tag{1.154}$$

It should be mentioned that the α and β-rules above are not really independent because of the consistency rule and of the fact that α and β-formulas dualize under negation. We also provide an alternative description of the rules.

Proposition 1.72. For an inference relation:

1. the α-rule is equivalent to the three conditions

$$\alpha \vdash \alpha_i \quad (i = 1, 2), \qquad \alpha_1, \alpha_2 \vdash \alpha; \tag{1.155}$$

2. The β-rule is equivalent to the three conditions

$$\beta_i \vdash \beta \quad (i = 1, 2), \qquad \beta, \neg\beta_1, \neg\beta_2 \vdash \bot. \tag{1.156}$$

Proof. ① Assume the α-rule holds. Since $\alpha \vdash \alpha$ by reflexivity, the α-rule yields $\alpha \vdash \alpha_i$. Also, since $\alpha_1, \alpha_2 \vdash \alpha_i$ for both $i = 1, 2$ again by assumption, the α-rule yields $\alpha_1, \alpha_2 \vdash \alpha$. Conversely, assume the conditions in formula (1.155) hold. Necessity of the α-rule is proved in the first table below, sufficiency in the second.

STEP	CLAIM	REASON
1	$F \vdash \alpha$	hypothesis
2	$\alpha \vdash \alpha_i$	hypothesis
3	$F \vdash \alpha_i$	1, 2, cut

STEP	CLAIM	REASON
1	$F \vdash \alpha_1$	hypothesis
2	$F \vdash \alpha_2$	hypothesis
3	$\alpha_1, \alpha_2 \vdash \alpha$	hypothesis
4	$F, \alpha_2 \vdash \alpha$	1, 3, cut
5	$F \vdash \alpha$	2, 4, cut

(1.157)

② Assume the β-rule holds. Since $\neg\beta, \beta \vdash \bot$ (proposition 1.68), from the β-rule we have $\neg\beta, \beta_i \vdash \bot$ and by consistency, $\beta_i \vdash \beta$. Also $\neg\beta_1, \neg\beta_2, \beta_i \vdash \bot$ (proposition 1.68) and hence $\neg\beta_1, \neg\beta_2, \beta \vdash \bot$ by the β-rule. Conversely, if conditions in formula (1.156) hold, then necessity and sufficiency of the β-rule are proved by the tables below.

STEP	CLAIM	REASON
1	$F, \beta \vdash \bot$	hypothesis
2	$\beta_i \vdash \beta$	hypothesis
3	$F, \beta_i \vdash \bot$	2, 1, cut

STEP	CLAIM	REASON
1	$F, \beta_1 \vdash \bot$	hypothesis
2	$F, \beta_2 \vdash \bot$	hypothesis
3	$\beta, \neg\beta_1, \neg\beta_2 \vdash \bot$	hypothesis
4	$F, \beta, \neg\beta_2 \vdash \bot$	1, 3, inconsistency
5	$F, \beta \vdash \bot$	2,4, inconsistency

(1.158)

The way complete regularity is used to prove completeness is via the concept of complete consistent set.

Definition 1.73. A consistent set F is *complete* if for every formula $\varphi \in L$ either $\varphi \in F$ or $\neg\varphi \in F$.

Observe that if F is consistent and the inference relation is regular, it can not happen that both $\varphi \in F$ and $\neg\varphi \in F$ (proposition 1.68).

Proposition 1.74. A consistent set $F \subseteq L$ is complete if and only if it is maximal among consistent sets with respect to inclusion.

Proof. ① Necessity. Assume F is complete. If $F \subset G$ there exists $\varphi \in G \setminus F$; since F is complete $\neg\varphi \in F$ and hence $\neg\varphi \in G$. Thus $\varphi, \neg\varphi \in G$ and G is inconsistent (proposition 1.68, 3). ② Sufficiency. Assume F is maximal among consistent sets. If $\varphi \notin F$ then $F \subset F \cup \{\varphi\}$ and $F, \varphi \vdash \bot$ by maximality of F. Replacing φ with $\neg\varphi$ we see that if $\neg\varphi \notin F$ then $F, \neg\varphi \vdash \bot$. Thus, if $\varphi, \neg\varphi \notin F$ then F is inconsistent (proposition 1.68, 1), a contradiction, and therefore either $\varphi \in F$ or $\neg\varphi \in F$ and F is complete. □

LEMMA 1.75. (LINDENBAUM) For a completely regular inference, every consistent set $F \subseteq L$ is contained in a complete consistent set.

Proof. Let P be the set of consistent sets containing F, ordered by inclusion. We claim that if

$$F_0 \subseteq F_1 \subseteq F_2 \subseteq \dots \tag{1.159}$$

is an ascending chain of elements of F, then $F_\infty = \bigcup_{n+0}^{\infty} F_n$ is an upper bound for the elements of the chain in P.

1. $F_n \subseteq F_\infty$ for every n by definition of F_∞.
2. $F \subseteq F_0$ because $F_0 \in P$ and $F_0 \subseteq F_\infty$ by 1, hence $F \subseteq F_\infty$.
3. F_∞ is consistent. Every finite subset $E \subseteq F_\infty$ must be contained in some F_n which is consistent, and therefore E is consistent. Since every finite subset of F_∞ is consistent, it follows that F_∞ is consistent.

This proves that every ascending chain in P has an upper bound in P and by the Zorn lemma P has a maximal element M. Being a maximal consistent set, M is complete (proposition 1.74) and being in P it contains F. □

PROPOSITION 1.76. If $\vdash$ is a completely regular inference and $F \subseteq L$ is a complete consistent set, then

$$\varphi \in F \Leftrightarrow F \vdash \varphi \qquad \varphi \notin F \Leftrightarrow F, \varphi \vdash \bot. \tag{1.160}$$

Proof. ① Necessity follows from the assumption rule. Sufficiency. Assume $F \vdash \varphi$. If $\varphi \notin F$ then $\neg\varphi \in F$ by completeness and $F \vdash \neg\varphi$ by assumption; from $F \vdash \varphi$ and $F \vdash \neg\varphi$ it follows that $F \vdash \bot$ (proposition 1.68), contradicting the consistency of F. Thus, if $F \vdash \varphi$ then $\varphi \in F$. ② Necessity is proved by the first table below. The second table shows that if $F, \varphi \vdash \bot$ and $\varphi \in F$, then F is inconsistent; thus $F, \varphi \vdash \bot$ implies $\varphi \notin F$, which is sufficiency.

STEP	CLAIM	REASON
1	$\varphi \notin F$	hypothesis
2	$\neg\varphi \in F$	1, completeness
3	$F \vdash \neg\varphi$	2, assumption
4	$\varphi \vdash \varphi$	assumption
5	$F, \varphi \vdash \bot$	3, 4, inconsistency general

STEP	CLAIM	REASON
1	$F, \varphi \vdash \bot$	hypothesis
2	$\varphi \in F$	hypothesis
3	$F \vdash \varphi$	2, part 1
4	$F \vdash \bot$	3, 2, cut

(1.161)

COROLLARY 1.77. If F is complete consistent and the inference relation is completely regular, the following properties hold.

1. If α is a conjunctive formula, then $\alpha \in F$ if and only if $\alpha_1 \in F$ and $\alpha_2 \in F$.
2. If β is a disjunctive formula, then $\beta \in F$ if and only if $\beta_1 \in F$ or $\beta_2 \in F$.

Proof. ① By proposition 1.76 and the α-rule, we have

$$\alpha \in F \Leftrightarrow F \vdash \alpha \Leftrightarrow F \vdash \alpha_1 \text{ and } F \vdash \alpha_2 \Leftrightarrow \alpha_1 \in F \text{ and } \alpha_2 \in F. \tag{1.162}$$

② Again from proposition 1.76 and the β-rule, we have

$$\beta \notin F \Leftrightarrow F, \beta \vdash \bot \Leftrightarrow F, \beta_1 \vdash \bot \text{ and } F, \beta_2 \vdash \bot \Leftrightarrow \beta_1 \notin F \text{ and } \beta_2 \notin F. \tag{1.163}$$

By negation, $\beta \in F$ if and only if $\beta_i \in F$ for at least one index i. □

Proposition 1.78. For a completely regular inference relation every consistent set is satisfiable.

Proof. Assume $F \subseteq L$ is consistent. By the Lindenbaum lemma, F is contained in a complete consistent set $\bar{F}$; if we prove that $\bar{F}$ is is satisfiable, then so is F. Thus, it suffices to prove that every complete consistent set F is satisfiable. Define a valuation $a : V \to \Omega$ setting $a(x) = 1 \Leftrightarrow x \in F$. We prove that the set $P = \{\varphi \in L : a(\varphi) = 1 \Leftrightarrow \varphi \in F\}$ coincides with L and we do this by structural induction with uniform notation. Observe first that $\varphi \in P \Rightarrow \neg\varphi \in P$; in fact, assuming $\varphi \in P$, by completeness of F we have

$$\neg\varphi \in F \Leftrightarrow \varphi \notin F \Leftrightarrow a(\varphi) = 0 \Leftrightarrow a(\neg\varphi) = 1 \tag{1.164}$$

and therefore $\neg\varphi \in P$. We now use structural induction.

1. Atomic case. By the above remark, it suffices to prove that $\varphi \in P$ for every atomic φ. If $\varphi = x$ is a variable, then $\varphi \in P$ by definition of a. If $\varphi = \bot$, then $\bot \notin F$ by consistency of F and $a(\bot) = 0$, hence $\bot \in P$. If $\varphi = \top$, then $F \vdash \top$ by the truth axiom and therefore $\top \in F$ (proposition 1.76); on the other habd, $a(\top) = 1$, so that $\top \in P$.
2. Double negation. This follows from the initial remark applied twice: if $\varphi \in P$ then $\neg\varphi \in P$ and therefore $\neg\neg\varphi \in P$.
3. Conjunctive formulas. Suppose $\varphi = \alpha$ and $\alpha_1, \alpha_2 \in P$. By proposition 1.76 and the semantics of conjunctive formulas we have

   $$\alpha \in F \Leftrightarrow \alpha_1, \alpha_2 \in F \Leftrightarrow a(\alpha_1) = a(\alpha_2) = 1 \Leftrightarrow a(\alpha) = 1 \tag{1.165}$$

   and therefore $\alpha \in P$.
4. Disjunctive formulas. Assume $\varphi = \beta$ and $\beta_1, \beta_2 \in P$. By completeness of F and properties of valuations we have

   $$\beta \in F \Leftrightarrow \beta_1 \in F \text{ or } \beta_2 \in F \Leftrightarrow a(\beta_1) = 1 \text{ or } a(\beta_2) = 1 \Leftrightarrow a(\beta) = 1 \tag{1.166}$$

   and therefore $\beta \in P$.

From proposition 1.78 and theorem 1.67, we finally have

Corollary 1.79. Every completely regular inference is complete. □

The whole point of this discussion is that in order to prove completeness of an inference relation we simply need to prove that it is completely regular, and this is considerably easier to do.

□ **Consistency classes.** We can replace complete regularity with a set of weaker axioms, which are even easier to check, although this requires further work. We set out to do that.

Definition 1.80. A *consistency class* for L is a set C of subsets $S \subseteq L$ satisfying the following conditions for every $S \in C$.

1. If $x \in S$ is a variable, then $\neg x \notin S$.
2. $\bot, \neg\top \notin S$.
3. If $\neg\neg\varphi \in S$, then $S \cup \{\varphi\} \in C$.
4. If $\alpha \in S$ is a conjunctive formula, then $S \cup \{\alpha_1, \alpha_2\} \in C$.
5. If $\beta \in S$ is a disjunctive formula, then $S \cup \{\beta_1\} \in C$ or $S \cup \{\beta_2\} \in C$.

Example 1.81 Every satisfiable formula generates a consistency class: consider for example the formula $\neg((x \vee y \to x) \to \neg y)$. A tableau for this formula is the following

$$\begin{tikzcd}[column sep=small]
 & \neg((x \vee y \to x) \to \neg y) \arrow[d, dash] & \\
 & x \vee y \to x \arrow[d, dash] & \\
 & \neg\neg y \arrow[d, dash] & \\
 & y \arrow[dl, dash] \arrow[dr, dash] & \\
\neg(x \vee y) \arrow[d, dash] & & x \\
\neg x \arrow[d, dash] & & \\
\neg y & &
\end{tikzcd}$$ (1.167)

Of the two branches of the tree, the left one is unsatisfiable because it contains both y and $\neg y$. The right branch, however, is satisfiable and the set of formulas in the branch is a consistency class whose elements are the sets

$$S_1 = \{\neg((x \vee y \to x) \to \neg y)\} \tag{1.168}$$
$$S_2 = \{\neg((x \vee y \to x) \to \neg y), x \vee y \to x, \neg\neg y\} \tag{1.169}$$
$$S_3 = \{\neg((x \vee y \to x) \to \neg y), x \vee y \to x, \neg\neg y, y\} \tag{1.170}$$
$$S_4 = \{\neg((x \vee y \to x) \to \neg y), x \vee y \to x, \neg\neg y, y, x\} \tag{1.171}$$

Note that we would equally well have obtained a consistency class by taking the set of all subsets of nodes from the branch; or the set whose elements are all the nodes of the branch. Or, we could have cut this at some node of the branch, take the nodes underneath and repeat the process.

Our aim is to prove that every set in a consistency class is satisfiable.

Definition 1.82. A *Hintikka set* is a set of formulas $H \subseteq L$ satisfying the following conditions:

1. If x is a variable and $x \in H$, then $\neg x \notin H$;
2. $\bot, \neg\top \notin H$;
3. If $\neg\neg\varphi \in H$, then $\varphi \in H$;
4. If $\alpha \in H$ then $\alpha_1, \alpha_2 \in H$;
5. If $\beta \in H$, then $\beta_1 \in H$ or $\beta_2 \in H$.

The relevance of the definition is the following

Lemma 1.83. (Hintikka) Every Hintikka set is satisfiable.

Proof. Suppose H is Hintikka. Define a valuation $a \in \Omega^V$ setting, for every variable x, $a(x) = 1 \Leftrightarrow x \in H$. We prove that $a(\varphi) = 1$ for every $\varphi \in H$ by induction on the degree d of φ. If $d = 0$ then either φ is a variable and in this case $a(\varphi) = 1$ be definition of a; or φ is a constant, in which case $\varphi = \top$ by definition of Hintikka set and $a(\varphi) = 1$. Assume now that $\varphi \in H$ has degree $d > 0$ and the statement is true for all formulas of H of degree less than d.

- If $\varphi = \neg\neg\psi$ then $\psi \in H$ by definition of Hintikka set; since ψ has degree less than d we have $a(\psi) = 1$ by the inductive hypothesis and therefore $a(\varphi) = a(\neg\neg\psi) = a(\psi) = 1$.
- If $\varphi = \alpha$, then $\alpha_1, \alpha_2 \in H$ by definition of Hintikka set and since both have degree less than d, $a(\alpha_1) = a(\alpha_2) = 1$; therefore $a(\alpha) = 1$.
- If $\varphi = \beta$, then $\beta_i \in H$ for at least one index i by definition of Hintikka set and since β_i has degree less than d, $a(\beta_i) == 1$; therefore $a(\beta) = 1$.

We show how consistency classes yield Hintikka sets. Recall that a set $S \in C$ is maximal if no element of C contains S properly, i.e. if $S \subseteq T \in C \Rightarrow S = T$.

PROPOSITION 1.84. If C is a consistency class, every maximal element $S \in C$ is a Hintikka set.

Proof. This is a straightforward application of the definition of consistency class and of maximal set.

- If $x \in S$ then $\neg x \notin S$ because C is a consistency class.
- $\bot, \neg\top \notin S$ because C is a consistency class.
- Suppose $\neg\neg\varphi \in S$. The $S \cup \{\varphi\} \in C$ because C is a consistency class. However, $S \subseteq S \cup \{\varphi\}$ and S is maximal in C, hence $S = S \cup \{\varphi\}$ and $\varphi \in S$.
- Suppose $\alpha \in S$. Since C is a consistency class, $S \cup \{\alpha_1, \alpha_2\} \in C$. But $S \subseteq S \cup \{\alpha_1, \alpha_2\}$ and S is maximal, hence $S = S \cup \{\alpha_1, \alpha_2\}$ and $\alpha_1, \alpha_2 \in S$.
- If βinS, the $S \cup \{\beta_i\} \in C$ for at least one index i because C is a consistency class. Since $S \subseteq S \cup \{\beta_i\}$ and S is maximal, $S = S \cup \{\beta_i\}$ and therefore $\beta_i \in S$.

The goal is now to show that every element S of a consistency class C is contained in a Hintikka set by showing that S is contained in a maximal element of C; unfortunately, although it is true that every element of a consistency class is satisfiable, it is not necessarily true that it is contained in a maximal element, so we need a detour.

DEFINITION 1.85. A consistency class C is *locally finite* when $S \in C$ if and only if every finite subset of S is in C.

Our plan is to prove that every consistency class is contained in one which is locally finite and that every element of a locally finite consistency class is contained in a maximal one which is Hintikka and therefore satisfiable.

PROPOSITION 1.86. Every consistency class C is contained in a locally finite consistency class D.

Proof. Declare that $S \in D$ if for every finite subset $U \subseteq S$ we can find $\bar{U}$ such that $U \subseteq \bar{U} \in C$. We prove that D is a consistency class.

- Suppose x is a variable and $x \in S \in D$. If $\neg x \in S$ then $U = \{x, \neg x\} \subseteq S$ is finite and therefore $\{x, \neg x\} \subseteq \bar{U} \in C$, contradicting the fact that C is a consistency class. Thus $\neg x \notin S$.
- If $S \in D$ and φ is either $\bot$ or $\neg\top$, then $\varphi \notin S$. For otherwise since $U = \{\varphi\} \subseteq S$ is finite, we would have $\{\varphi\} \subseteq \bar{U} \in C$, contradicting the fact that C is a consistency class.
- Suppose $\alpha \in S \in D$. If $U \subseteq S \cup \{\alpha_1, \alpha_2\}$ is finite, then $U \subseteq V \cup \{\alpha_1, \alpha_2\}$ where V is a finite subset of S containing α. Therefore, $\alpha \in \bar{V} \in C$ and since C is a consistently class $\bar{U} := \bar{V} \cup \{\alpha_1, \alpha_2\} \in C$. But then $U \subseteq \bar{U} \in C$ and therefore $S \cup \{\alpha_1, \alpha_2\} \in D$.
- Suppose $\beta \in S \in D$. If $S \cup \{\beta_1\}, S \cup \{\beta_2\} \notin D$, there exist two finite subsets $U_i \subseteq S \cup \{\beta_i\}$, $i = 1, 2$ which are not contained in an element of C. Observe that $\beta_i \in U_i$, for otherwise U_i, being a finite subset of S, would be contained in an element of C. Now observe that there exists a finite subset $U \subseteq S$ such that $\beta \in U$ and $U_1 \cup U_2 \subseteq U \cup \{\beta_1, \beta_2\}$. Since $U \subseteq S \in D$ is finite, there exists $\bar{U}$ such that $U \subseteq \bar{U} \in C$ and since $\beta \in \bar{U}$ and C is a consistency class, $\bar{U} \cup \{\beta_i\} \in C$ for some index i. But then $U_i \subseteq \bar{U} \in C$ against the assumption.

To see that $C \subseteq D$, observe that given any $S \in C$, every finite subset $U \subseteq S$ is contained in $S \in C$ and therefore $S \in D$ by definition of D.
We prove that D is locally finite. If $S \in D$ and $T \subseteq S$ is finite, then every finite subset $U \subseteq T$ is also a finite subset of S and therefore there exists $\bar{U}$ such that $U \subseteq \bar{U} \in S$; thus $T \in D$ — note that finiteness of T does not play any role here. Conversely, suppose $U \in D$ for every finite subset $U \subseteq S$. Since U is a finite subset of itself, $U \subseteq \bar{U} \in C$ and $S \in D$. □

Observe that the last part of the proof also shows that if C is locally finite, $S \in C$ and $T \subseteq S$, then $T \in C$; in other words, every locally finite consistency class is downward closed.

Proposition 1.87. If C is a locally finite consistency class, then every $S \in C$ is contained in a maximal element of C.

Proof. Let $P = \{T \in C : S \subseteq T\}$ the the set of elements of C containing S, partially ordered by inclusion. We prove that P is inductive. Suppose

$$S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots \tag{1.172}$$

is an ascending chain in P. We claim that $S_\infty = \bigcup_{n=0}^\infty S_n \in C$. For suppose $U \subseteq S_\infty$ is finite; then every $x \in U$ must belong to some S_i and therefore $U \subseteq S_n$ for some n; hence $U \in C$ because C is downward closed and therefore $S_\infty \in C$ because C is locally finite. Now clearly $S \subseteq S_\infty$ and hence $S_\infty \in P$. Also S_∞ is an upper bound for the chain. Thus, P is inductive and by the Zorn lemma it has a maximal element $\bar{S}$ which is also a maximal element of C containing S. □

Corollary 1.88. Every element of a consistency class is satisfiable.

Proof. Suppose C is a consistency class and $S \in C$. C is contained in a locally finite consistency class D (proposition 1.86) and $S \in D$. Now S is contained in maximal element $\bar{S}$ of D (proposition 1.87) which is a Hintikka set (proposition 1.86) and therefore satisfiable (lemma 1.83). However, if $a \vDash \bar{S}$, then $a \vDash S$. □

□ **Regular inference.** We can provide a weaker alternative to complete regularity.

Definition 1.89. An inference relation is *regular* if it satisfies the following conditions.

$$F \vdash \alpha \;\Rightarrow\; F \vdash \alpha_1 \text{ and } F \vdash \alpha_2 \qquad \alpha\text{-formulas} \tag{1.173}$$

$$F, \beta_1 \vdash \bot \text{ and } F, \beta_2 \vdash \bot \;\Rightarrow\; F, \beta \vdash \bot \qquad \beta\text{-formulas} \tag{1.174}$$

$$F \vdash \top \qquad \text{truth} \tag{1.175}$$

Much like for complete regularity, we can replace the original definition with slightly simplified rules

Proposition 1.90. An inference relation is regular if and only if it satisfies the following conditions.

1. $\alpha \vdash \alpha_i$ for $i = 1, 2$.
2. $\beta, \neg\beta_1, \neg\beta_2 \vdash \bot$
3. $\vdash \top$

Proof. ① Suppose (1.173) holds. Setting $F = \{\alpha\}$, since $\alpha \vdash \alpha$ by assumption, we have $\alpha \vdash \alpha_i$. Conversely if condition 1 holds and $F \vdash \alpha$, the $F \vdash \alpha_i$ by the cut rule. ② Suppose (1.174) holds. Set $F = \{\neg\beta_1, \neg\beta_2\}$; then $F, \beta_i \vdash bottom$ by proposition 1.68 and therefore $\beta, \neg\beta_1, \neg\beta_2 \vdash \bot$ by (1.174). Conversely, if consition 2 holds and $F, \beta_i \vdash \bot$, then $F \vdash \bot$ as in the second derivation of (1.158). ③ If (1.158) holds for every F, it holds in particular when $F = \varnothing$ and therefore $\vdash \varphi$. Conversely, if condition 3 holds, (1.158) follows by weakening. □

The following is the key fact about regularity.

Proposition 1.91. For a regular inference, the class

$$C = \{F \subseteq L : F \nvdash \bot\} \tag{1.176}$$

of consistent sets is a consistency class.

Proof. We verify that the conditions in the definition of consistency class are satisfied.

- If $x, \neg x \in F$ then $F \vdash \bot$ (proposition 1.68). Thus if $x \in F \in C$, then $\neg x \notin F$.
- If $\bot \in F$ then $F \vdash \bot$ by assumption; thus, if $F \in C$, $\bot \notin F$. Also, if $\neg\top \in F$ then $F \vdash \neg\top$ by assumption; by the truth rule $F \vdash \top$ and therefore $F \vdash \bot$ (proposition 1.68); thus $\neg\top \notin F$.
- Double negation. Assume $\neg\neg\varphi \in F$. The proof below shows that if $F \cup \{\varphi\} \notin C$, then F is inconsistent. Therefore, if $\neg\neg\varphi \in F \in C$, then $F \cup \{\varphi\} \in C$.

Step	Claim	Reason
1	$\neg\neg\varphi \in F$	hypothesis
2	$F, \varphi \vdash \bot$	hypothesis
3	$F \vdash \neg\neg\varphi$	1, assumption
4	$F, \neg\neg\varphi \vdash \varphi$	formula (1.150)
5	$F \vdash \varphi$	3, 4, cut
6	$F \vdash \bot$	5, 2, cut

(1.177)

- α-formulas. Assume $\alpha \in F$. The derivation on the left below shows that if $F \cup \{\alpha_1, \alpha_2\} \notin C$, then F is inconsistent. Therefore, if $\alpha \in F \in C$, then $F \cup \{\alpha_1, \alpha_2\} \in C$.

Step	Claim	Reason
1	$\alpha \in F$	hypothesis
2	$F, \alpha_1, \alpha_2 \vdash \bot$	hypothesis
3	$F \vdash \alpha$	1, assumption
4	$F \vdash \alpha_1$	3, α-rule
5	$F, \alpha_2 \vdash \bot$	4, 2, cut
6	$F \vdash \alpha_2$	3, α-rule
7	$F \vdash \bot$	6, 5, cut

Step	Claim	Reason
1	$\beta \in F$	hypothesis
2	$F, \beta_1 \vdash \bot$	hypothesis
3	$F, \beta_2 \vdash \bot$	hypothesis
4	$F \vdash \beta$	1, assumption
5	$F, \beta \vdash \bot$	2, 3, β-rule
6	$F \vdash \bot$	4, 5, cut

(1.178)

- β-formulas. Assume $\beta \in F$. The derivation on the right above shows that if $F \cup \{\beta_i\} \notin C$ for $i = 1, 2$, then F is inconsistent. Therefore, if $\beta \in F \in C$, then $F \cup \{\beta_i\} \in C$ for at least one i.

We therefore have an alternative way of proving completeness for an inference relation:

Corollary 1.92. Every regular inference relation is complete.

Proof. If $\vdash$ is a regular inference, the the class C of consistent sets is a consistency class (proposition 1.91). Therefore, every consistent set is satisfiable (and the inference is complete (theorem 1.67). □

The advantage of regularity over complete regularity is, of course, that it is easier to check.

1.5 The tableau calculus

□ **Notation.** The calculus uses $C = \{\bot, \top, \neg, \wedge, \vee, \rightarrow\}$ as the set of connectives; biimplication is regarded as the conjunction of two implications. The tableau calculus concerns trees of formulas and defines inference (or expansion) rules that allow to modify these trees. We start with these rules.

Definition 1.93. The inference or expansion rules for tableaux are the following.

1. True and false. If $\neg\top$ occurs in a node of a branch B, expand B adding $\bot$ at the end of the branch. Dually, if $\neg\bot$ occurs in B, expand B with $\top$.
2. Double negation. If the formula $\neg\neg\varphi$ occurs in a node of a branch B, add φ at the end of the branch.
3. α-expansion. If a conjunctive formula α occurs in a branch B, extend B with any of the α_i.
4. β-expansion. If a disjunctive formula β occurs in a branch B, add two children at the end of B with nodes β_1 and β_2

Note that applying α-expansion twice in a row it is possible to add both α_1 and α_2 to every branch containing α . Likewise, repeated application of β expansion allows to fork with β_1 and β_2 every branch containing β.

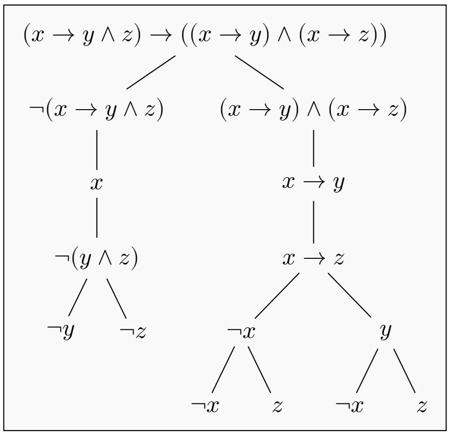
Definition 1.94. Given a set of formulas $F \subseteq L$, a *tableau expansion* of F is any tree redursively defined by the following rules: the single branch tree

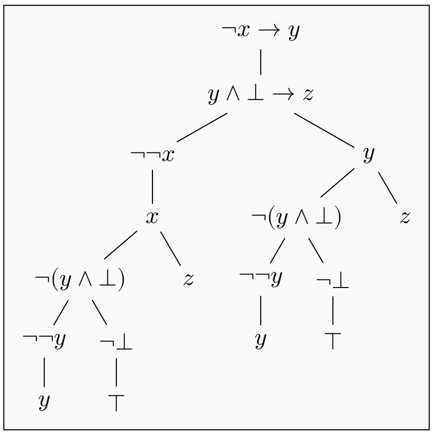
$$\begin{array}{c} \varphi_1 \\ | \\ \varphi_2 \\ \vdots \\ \varphi_n \end{array} \tag{1.179}$$

where $E = \{\varphi_1, \dots, \varphi_n\} \subseteq F$ is a finite subset, is a tableau expansion of F and the tree obtained brom a tableau expansion using a tableau expansion rule is also a tableau expansion of F. If F is infinite, a tableau expansion of F is a tableau expansion of any finite subset of F.

We can think of a semantic tableau as the disjunction of its branches, with each branch beaing the conjunction of the formulas in ots nodes.

Example 1.95 A semantic tableau for $(x \to y \wedge z) \to ((x \to y) \wedge (x \to z))$ is given in the first tree of (1.180). The formula at the root is a β-formula, so we use β-expansion at height 1 and branch the tree adding β_1 on the left and β_2 on the right. In the left branch, $\neg(x \to y \wedge z)$ is an α-formula, so we use α expansion twice: first adding $\alpha_1 = x$ to the branch, then adding $\alpha_2 = \neg(y \wedge z)$. the latter is again a β-formula and an application of the β-rule provides another branching of the tree with the β_i's. Again at height 1 in the right branch from the root we have another α-formula and use α-expansione twice. At this point we use β-expansion on $x \to y$ and obtain a branching point and then apply β-expansion again to $x \to z$ to both nodes just obtained.
Note that we have completely expanded the tree and no further expansion is possible, unless we decide to repeat one of the expansions we already used, which is possible although useless. Note also that we could have stopped at any intermediate step and the result would have still been a tableau for the original formula.





(1.180)

Example 1.96 A tableau for the set $F = \{\neg x \to y,\ y \wedge \bot \to z\}$ is given in the second tree of (1.180). Note that we start by writing in sequence, as in the case of α-expansion, the formulas of F.

Both tableaux in examples 1.95 and 1.96 have an important feature, that all expansion rules have been applied to formulas in every branch. We make this more precise.

Definition 1.97. A branch B of a tableau is *complete* if it satisfies the following conditions.

- If $\neg\top$ appears in B so does $\bot$ and if $\neg\bot$ appears in B so does $\top$.
- If a double negation $\neg\neg\varphi$ appears in B, so does φ.
- If an α-formula appears in B, so do α_1 and α_2.
- If a β-formula appears in B, so does at least one of β_1 or β_2.

□ **The inference induced by tableaux.** We now show how tableaux can be used to define an inference relation that will turn out to be sound and complete.

Definition 1.98. A branch B of a tableau is *closed* if it contains either $\bot$ or a pair $\{\varphi, \neg\varphi\}$ consisting of a formula and its negation. A Tableau is *closed* if every branch is closed.

Observe that we may not need to completely expand a tableau to verify that it is closed, for as soon as we meet a pair $\{\varphi, \neg\varphi\}$ or $\bot$ in a branch B, that branch and all branches it may generate by β-expansion will be closed.

Definition 1.99. Given $F \subseteq L$ and $\varphi \in L$, we say that φ can be inferred from F if $F \cup \{\neg\varphi\}$ has a closed tableau. In this case, we write

$$F \vdash \varphi. \tag{1.181}$$

Example 1.100 We will use a semantic tableau to prove that

$$x \to y,\ \ y \to z \vdash x \to z. \tag{1.182}$$

A closed tableau for $F \cup \{\neg\varphi\} = \{x \to y,\ \ y \to z,\ \ \neg(x \to z)\}$ is shown below. The tableau starts with the three assumptions; we then use α-expansion on the third assumption, writing both α_1 and α_2. Then we use β-expansion on the first assumption producing two branches, starting with $\neg x$ and y. Observe that the left branch is already closed, so that we need only consider the right branch. If we use β-expansion on the second assumption we now obtain two closed branches and the tableau is closed, thus proving the claim.

$$\begin{tikzcd}
& x \to y \arrow[d, dash] & & \\
& y \to z \arrow[d, dash] & & \\
& \neg(x \to z) \arrow[d, dash] & & \\
& x \arrow[d, dash] & & \\
& \neg z \arrow[dl, dash] \arrow[dr, dash] & & \\
\neg x & & y \arrow[dl, dash] \arrow[dr, dash] & \\
& \neg y & & z
\end{tikzcd} \tag{1.183}$$

This example shows that the tableau inference is non-deterministic: we might have chosen to use β-expansion of the first assumption to start with, and we would have obtained a different, yet still closed tableau.

Lemma 1.101. $F \subseteq L$ is inconsistent if and only if it has a closed tableau.

Proof. Recall that F is inconsistent if $F \vdash \bot$, i.e. if $F \cup \{\neg\bot\}$ has a closed tableau. Thus, the claim amounts to the statement that F has a closed tableau if and only if $F \cup \{\neg\bot\}$ does. Next, observe that if $F \subseteq G$ are sets of formulas, then every tableau for F is also a tableau for G simply because every finite subset of F is also a subset of G. Thus, if F has a closed tableau, the so does $F \cup \{\neg\bot\}$. Conversely, suppose $F \cup \{\neg\bot\}$ has a closed tableau with a finite set of assumptions E. Since the only expansion of $\neg\bot$ is $\top$, a branch B of a tableau in which $\neg\top$ occurs can only be closed if $\bot$ or another pair $\{\varphi, \neg\varphi\}$ occurs in B. Thus, $\neg\bot$ can be omitted from E and we now have a closed tableau for F. □

Proposition 1.102. The tableau calculus induces an inference relation on L.

Proof. ① Compactness. This follows from the fact that a closed tableau of $F \cup \{\neg\varphi\}$ is, by definition, a closed tableau for finite subset $E \cup \{\neg\varphi\}$. ② Assumption. If $\varphi \in F$, then below is a closed tableau for $F \cup \{\neg\varphi\}$, so that $F \vdash \varphi$

$$\begin{tikzcd}
\varphi \arrow[d, dash] \\
\neg\varphi
\end{tikzcd} \tag{1.184}$$

③ Cut. Assume $F, \varphi \vdash \bot$ and $F, \neg\varphi \vdash \bot$. Then both $F \cup \{\varphi\}$ and $F \cup \{\neg\varphi\}$ have closed tableaux (lemma 1.101). If F has an open tableau it must have a satisfiable branch and must be thus satisfiable, say $a \vDash F$. But the either $a \vDash \varphi$ or $a \vDash \neg\varphi$ implies that either $F \cup \{\varphi\}$ or $F \cup \{\neg\varphi\}$ are satisfiable and can not have any closed tablea, a contradiction; thus F has a closed tableau and is inconsistent. Now cut follows from proposition 1.68. ④ Consistency. It suffices to observe that $F \cup \{\neg\varphi\}$ is inconsistent if and only if it has a closed tableau (lemma 1.101), i.e. if and only if $F \vdash \varphi$. □

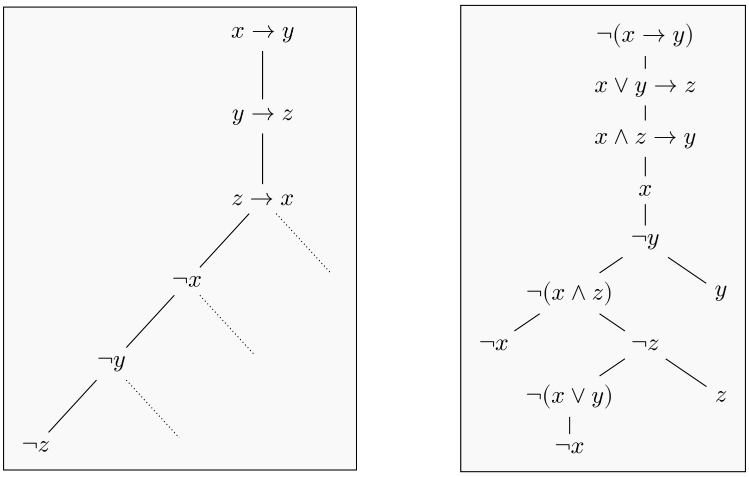
□ **Soundness.**

Definition 1.103. A branch of a tableau is *satisfiable* or *open* if there exists a valuation satisfying all the formulas of the branch. A tableau is *satisfiable* or *open*if it has at least one satisfiable branch.

Example 1.104 We saw in example 1.30 that $F = \{x \to y,\ y \to z,\ z \to x\}$ is satisfiable using a truth table. We will now show that it has a an open tableau. Note that in order to do this, we actually only need to construct an open branch of a tableau and this can be done as shown in the first tableau of (1.186). The node $\neg x$ at height 3 is the β_1 component from the β-expansion of the root and the other nodes on the left diagonal are likewise obtained from β-expansion of the other formulas in F. This branch is complete and does not contain neither $\bot$ nor a negated pair and therefore is open. Since $\neg x, \neg y, \neg z \in B$, the only valuation $a \in \Omega^V$ that satisfies this branch is defined by

$$a(x) = a(y) = a(z) = 0. \tag{1.185}$$

Thus, $a \vDash F$ and F is satisfiable.



(1.186)

Example 1.105 Observe that a closed branch is necessarily unsatisfiable; therefore a closed tableau is unsatisfiable. An example of such a tableau for

$$F = \{\neg(x \to y), x \vee y \to z, x \wedge z \to y\}. \tag{1.187}$$

is shown in the second tree of (1.186). We will see later that if a set has a closed tableau, then all its tablaeaux are closed and therefore it can not have an open one.

Lemma 1.106. An application of a tableau expansion rule to a satisfiable tableau produces a satisfiable tableau. In particular, any satisfiable set can not have a closed tableau.

Proof. Suppose B is a satisfiable branch of a tableau T and $a \vDash B$. We prove that any expansion rule applied to B produces a satisfiable branch. ① True and false. Observe first that $\neg\top$ can not occur in B, because B is satisfiable. If $\neg\bot$ occurs in B, then $a \vDash B \cup \{\top\}$ which is the expansion of B. ② Double negation. If $\neg\neg\varphi \in B$ then $a \vDash \neg\neg\varphi$; therefore $a \vDash \varphi$ and hence $a \vDash B \cup \{\varphi\}$. ③ α-expansion. If $\alpha \in B$ then $a \vDash \alpha$; therefore $a \vDash \alpha_i$ and therefore $a \vDash B \cup \{\alpha_i\}$. ④ β-expansion. If $\beta \in B$, then $a \vDash \beta$ and therefore either $a \vDash \beta_1$ or $a \vDash \beta_2$; say that $a \vDash \beta_i$. Then $a \vDash B \cup \{\beta_i\}$ which is a branch of T and thus T is still open. If F is satisfiable, its only branch is satisfiable and any application of an expansion rule will produce at lest one satisfiable branch. □

Theorem 1.107. (Soundness) The tableau calculus is sound:

$$F \vdash \varphi \ \Rightarrow\ F \vDash \varphi. \tag{1.188}$$

Proof. If $F \vdash \varphi$ then $F \cup \{\neg\varphi\}$ has a closed tableau. This implies that $F \cup \{\neg\varphi\}$ is unsatisfiable (lemma 1.106) and therefore $F \vDash \varphi$. □

□ **Completeness.**

PROPOSITION 1.108. The tableau inference relation is regular.

Proof. The following tableaux prove the claim of proposition 1.90 and therefore prove regularity.

$$\begin{tikzcd}
 & & & \beta \arrow[d, dash] & \\
\neg\top \arrow[d, dash] & \alpha \arrow[d, dash] & & \neg\beta_1 \arrow[d, dash] & \\
\bot & \neg\alpha_i \arrow[d, dash] & & \neg\beta_2 \arrow[dl, dash] \arrow[dr, dash] & \\
 & \alpha_i & \beta_1 & & \beta_2
\end{tikzcd} \tag{1.189}$$

COROLLARY 1.109. The tableau calculus is complete.

1.6 Resolution

Resolution is a calculus amenable to automatic computation, although the proofs produced are hardly of immediate interpretation.

□ **Notation.** When dealing with resolution we use $C = \{\bot, \top, \neg, \wedge, \vee, \rightarrow\}$ as the set of connectives and regard biimplication as an abbreviation for the conjunction of two implications.
In addition, the resolution calculus uses a peculiar notation, derived from conjunctive normal forms. Recall that a generalized clause is a disjunction of arbitrary formulas, called the components of the clause. We use set-theoretical notation and write a generalized clause $\varphi = \psi_1 \vee \cdots \vee \psi_n$ as a set $\varphi = \{\psi_1, \ldots, \psi_n\}$ whose elements are the *components* of φ; we say that these components *occur* in the generalized clause. Note that we will not distinguish between the logical and the set theoretical notation. Note also that the set-theoretical notation excludes duplicate components and therefore implies that the generalized clauses are in reduced form. We will use the term clause to include generalized clauses. We regard any formula as a conjunction of generalized clauses; this is always possible, even if the outermost connective is not a conjunction, because we can regard the formula as a conjunction of a single element, the formula itself. We extend the set theoretical notation to represent formulas as the set of their conjuncts. Thus, a formula is represented as a set whose elements are generalized clauses, i.e. sets of components.

Example 1.110 Consider the formula

$$[(x \rightarrow z) \vee (y \rightarrow z)] \wedge [(x \wedge y \rightarrow z) \vee (x \wedge z) \vee (x \vee z \rightarrow y)]\,. \tag{1.190}$$

This formula is a conjunction of two generalized clauses, written in square brackets: the first clause is the disjunction of two components, the second of three components. We therefore have a set with two elements, each corresponding to a generalised clause; the first element is a set with two elements, the components f the first clause; the seconnd element is a set with three elements, the components of the second clause. The set-theoretical version of the formula is therefore

$$\{\{x \rightarrow z,\ \ y \rightarrow z\},\quad \{x \wedge y \rightarrow z,\ \ x \wedge z,\ \ x \vee z \rightarrow y\}\}\,. \tag{1.191}$$

When dealing with sets of formulas, we regard them as a conjunction of formulas, mainly because we are interested in examining the inference relation $F \vdash \varphi$ induced by the resolution calculus and in this context F can be thought of as the conjunction of its elements. Thus, we regard F as a set of generalized clauses, those coming from all the formulas of F.

Example 1.111 The set of formulas

$$\{(x \vee (y \to z)) \wedge ((x \to y) \vee (x \to z)),\ x \wedge (y \leftrightarrow z),\ x \vee (\neg y \wedge \neg z)\} \tag{1.192}$$

translates in clausal form into the set

$$\{\{x, y \to z\},\ \{x \to y, x \to z\},\ \{x\},\ \{y \leftrightarrow z\},\ \{x, \neg y \wedge \neg z\}\} \tag{1.193}$$

□ **The calculus and its inference.** The resolution calculus is assigned by a set of axioms and a set of inference rules which then combine to give an inference relation.

Definition 1.112. The resolution calculus has two axiom schemes.

- ○ Constants: the generalized clauses $\{\top\}$ and $\{\neg\bot\}$ are axioms.
- ○ Disjunction: the generalized clause $\{\varphi, \neg\varphi\}$ is an axiom for every $\varphi \in L$.

Axioms do not actually play any significant role when checking inference in concrete cases, although they have a use in the theoretical aspects of the theory. Inference rule describe how to derive new clauses from a given set.

Definition 1.113. The following inference rules, in which φ, φ_i and φ_j are generalized clauses, are allowed in the resolution calculus.

- ○ Falsehood. From a clause φ in which $\bot$ occurs, one can derive $\varphi[\varnothing/\bot]$ by omitting the occurrence of $\bot$. Likewise, if $\neg\top$ occurs in φ one can derive $\varphi[\varnothing/\neg\top]$ by omitting $\neg\top$.
- ○ Double negation. From a clause φ in which If $\neg\neg\psi$ occurs, one can derive $\varphi[\psi/\neg\neg\psi]$ by replacing the occurrence of $\neg\neg\psi$ with ψ.
- ○ α-expansion. From a clause φ in which a conjunctive formula α occurs, one can derive $\varphi[\alpha_i/\alpha]$ by replacing the occurrence of α with either α_1 or α_2.
- ○ β-expansion. From a clause φ in which a disjunctive formula β occurs, one can derive the clause $\varphi[\beta_1, \beta_2/\beta]$ by replacing β with the two components β_1 and β_2.
- ○ Resolution. If the component ψ occurs in φ_i and $\neg\psi$ occurs in φ_j, one can infer the clause $\varphi_i[\varnothing/\psi] \cup \varphi_j[\varnothing/\neg\psi]$ by omitting ψ from φ_i, $\neg\neg\psi$ from φ_j and then forming the union of the remaining components.

The inference rules can be represented schematically as follows; the clauses appearing on the top of each rule are its premises and the clause at the bottom is the conclusion.

$\frac{\varphi_i}{\varphi_i[\varnothing/\bot]}$	$\bot \in \varphi_i$	$\frac{\varphi_i}{\varphi_i[\varnothing/\neg\top]}$	$\neg\top \in \varphi_i$	$\frac{\varphi_i}{\varphi_i[\psi/\neg\neg\psi]}$	$\neg\neg\psi \in \varphi_i$
$\frac{\varphi_i}{\varphi_i[\alpha_j/\alpha]}$	$\alpha \in \varphi_i$	$\frac{\varphi_i}{\varphi_i[\beta_1,\beta_2/\beta]}$	$\beta \in \varphi_i$	$\frac{\varphi_i \quad \varphi_j}{\varphi_i[\varnothing/\psi] \cup \varphi_j[\varnothing/\neg\psi]}$	$\psi \in \varphi_i,\ \neg\psi \in \varphi_j$

(1.194)

Note that in the α-rule one can infer two generalized clauses, one for every value of the index i.
Also, note that the resolution rule is not really meaningful when $\varphi_i = \varphi_j = \varphi$, i.e. when the two clauses coincide. For in this case, both ψ and $\neg\psi$ occur in φ and $\varphi[\varnothing/\psi] \cup \varphi[\varnothing/\neg\psi] = \varphi$ so that we are back at the starting point.

Definition 1.114. An *expansion* of a set $F \subseteq L$ of formulas, regarded as a set of generalized clauses, is a finite sequence $S = (\varphi_1, \dots, \varphi_n)$ of generalized clauses in which every φ_i is obtained from one of the following rules.

- ○ Assumption: $\varphi_i \in F$
- ○ Axiom: φ_i is an axiom.
- ○ Inference: φ_i is the conclusion of an inference rule whose premises are clauses of S preceding φ_i.

An expansion is *closed* if one of its closes is $\bot$.

Example 1.115 We prove that the set of formulas $F = \{x \leftrightarrow y,\ y \to z,\ x \wedge \neg z\}$ has a closed expansion.

Step	Clause	Rule
1	$\{x \leftrightarrow y\}$	assumption
2	$\{y \to z\}$	assumption
3	$\{x \wedge \neg z\}$	assumption
4	$\{x \to y\}$	1, α-expansion
5	$\{\neg x, y\}$	4, β-expansion
6	$\{\neg y, z\}$	2, β-expansion

Step	Clause	Rule
7	$\{x\}$	3, α-expansion
8	$\{\neg z\}$	3, α-expansion
9	$\{y\}$	5, 7, resolution
10	$\{z\}$	6, 9, resolution
11	$\varnothing$	8, 10, resolution

(1.195)

Observe that in row 4 we have used α-expansion because biimplication is defined, for the purposes of resolution, as the conjunction of two implications, so that $x \leftrightarrow y$ really means $(x \to y) \wedge (y \to x)$; also note that we have only used the α_1-part of the expansion simply because the α_2-part is not needed.

Definition 1.116. The *inference relation* induced by the resolution calculus is defined by stipulating that given a set of formulas $F \subseteq L$ and a formula $\varphi \in L$, $F \vdash \varphi$ if $F \cup \{\varphi\}$ has a closed expansion.

We think of a derivation as a conjunction of formulas; since every formula is in turn a conjunction of generalized clauses, we can think of a derivation as a conjunction of clauses. Forthis reason the inference rules are always among clauses.

Example 1.117 We prove that $x \wedge y \to z,\ \ x \to y \vdash x \to z$ by showing that the set $F = \{x \wedge y \to z,\ \ x \to y, \neg(x \to z)\}$ has a closed expansion.

Step	Clause	Rule
1.	$\{x \wedge y \to z\}$	assumption
2.	$\{x \to y\}$	assumption
3.	$\{\neg(x \to z)\}$	assumption
4.	$\{\neg(x \wedge y), z\}$	1, β-expansion
5.	$\{\neg x, \neg y, z\}$	4, β-expansion
6.	$\{\neg x, y\}$	2, β-expansion

Step	Clause	Rule
7.	$\{x\}$	3, α-expansion
8.	$\{\neg z\}$	3, α-expansion
9.	$\{\neg y, z\}$	5, 7, resolution
10.	$\{y\}$	6, 7, resolution
11.	$\{z\}$	9, 10, resolution
12.	$\varnothing$	8, 11, resolution

(1.196)

Note that asserting that φ is a syntactical consequence of F does not mean that φ appears in an expansion of F. In fact it is easy to see that if φ appears in an expansion of F, then certainly $F \vdash \varphi$ because $\neg\varphi \in F \cup \{\neg\varphi\}$ and since both φ and $\neg\varphi$ occur in an expansion of $F \cup \{\neg\varphi\}$, an application of the resolution rule yields a closed expansion. The converse, however, is not true: φ can be a syntactical consequence of F without appearing in any expansion of F.

Example 1.118 Observe that $x \vdash x \vee y$ because we have a closed expansion of $\{x, \neg(x \vee y)\}$:

Step	Clause	Rule
1	$\{x\}$	assumption
2	$\{\neg(x \vee y)\}$	assumption
3	$\{\neg x\}$	2, α-expansion
4	$\varnothing$	1,3, resolution

(1.197)

However, the only expansion of x is x itself, so that $x \vee y$ can not appear in any expansion of x.

☐ **Soundness.** We now prove that the calculus is sound, i.e. that the induced inference is sound.

THEOREM 1.119. The resolution calculus is sound:

$$F \vdash \varphi \quad \Rightarrow \quad F \vDash \varphi. \tag{1.198}$$

Proof. We first prove that the inference rules of the calculus (definition 1.113) are sound, in the sense that if a valuation a satisfies the premises of a rule in the formulas of (1.194), then it also satisfies the conclusion.

- Falsehood. If $\bot \in \varphi_i$, then $\varphi_i \equiv \bot \vee \chi$ where χ is a generalized clause not containing $\bot$. By the substitution lemma,

$$\varphi_i[\varnothing/\bot] \equiv (\bot \vee \chi)\,[\varnothing/\bot] = \chi \equiv \bot \vee \chi \equiv \varphi_i \tag{1.199}$$

  and therefore if $a \vDash \varphi_i$, then $a \vDash \varphi[\varnothing/\bot]$. The case for $\neg\top$ is identical, because $\neg\top \equiv \bot$.
- Double negation. If $\neg\neg\psi \in \varphi_i$ then $\varphi_i \equiv \neg\neg\psi \vee \chi$ where χ is a generalized clause not containing $\neg\neg\psi$. By the substitution lemma,

$$\varphi_i[\psi/\neg\neg\psi] \equiv (\neg\neg\psi \vee \chi)\,[\psi/\neg\neg\psi] = \psi \vee \chi \equiv \varphi_i \tag{1.200}$$

  and therefore if $a \vDash \varphi_i$, then $a \vDash \varphi[\psi/\neg\neg\psi]$.
- α-expansion. If $\alpha \in \varphi_i$ then $\varphi_i \equiv \alpha \vee \chi$ where χ is a generalised clause not containing α,

$$\varphi_i[\alpha_1/\alpha] \wedge \varphi_i[\alpha_2/\alpha] \equiv (\alpha_1 \vee \chi) \wedge (\alpha_2 \vee \chi) \equiv (\alpha_1 \wedge \alpha_2) \vee \chi \equiv \varphi_i \tag{1.201}$$

  and therefore if $a \vDash \varphi_i$, then $a \vDash \varphi_i[\alpha_j/\alpha]$ for $j = 1, 2$.
- β-expansion. If $\beta \in \varphi_i$ then $\varphi_i \equiv \beta \vee \psi$ where ψ is a generalised clause not containing β. Therefore,

$$\varphi_i[\beta_1, \beta_2/\beta] \equiv (\beta_1 \vee \beta_2) \vee \psi \equiv \beta \vee \psi \equiv \varphi_i \tag{1.202}$$

  and if $a \vDash \varphi_i$, then $a \vDash \varphi_i[\beta_1, \beta_2/\beta]$.
- Resolution. Assume $\psi \in \varphi_i$, $\neg\psi \in \varphi_j$ and $a \vDash \varphi_i, \varphi_j$. Then $\varphi_i \equiv \psi \vee \psi_i$ and $\varphi_j \equiv \neg\psi \vee \psi_j$ where ψ_i and ψ_j are generalized clauses with $\psi \notin \psi_i$ and $\neg\psi \notin \psi_j$, so that $\varphi_i[\varnothing/\psi] \cup \varphi_j[\varnothing/\neg\psi] \equiv \psi_i \vee \psi_j$. There are two possibilities:
  - if $a \vDash \psi$ then $a \nvDash \neg\psi$ and since $a \vDash \varphi_j$, we must have $a \vDash \psi_j$ and therefore $a \vDash \varphi_i[\varnothing/\psi] \cup \varphi_j[\varnothing/\neg\psi]$;
  - if $a \nvDash \psi$ then, since $a \vDash \varphi_i$, we must have $a \vDash \psi_i$ and therefore $a \vDash \varphi_i[\varnothing/\psi] \cup \varphi_j[\varnothing/\neg\psi]$.

Observe that every axiom is trivially satisfied by any valuation. As a consequence, if $F \subseteq L$ is a set of formulas and $a \vDash F$ then a also satisfies every expansion of F. Thus, if F has a closed expansion then it is usatisfiable, because no valuation can sitisfy the empty clause. Thus, if $F \vdash \varphi$ then $F \cup \{\neg\varphi\}$ has a closed expansion and is therefore unsatisfiable; therefore, $F \vDash \varphi$. □

Going back to the resolution inference rule, recall that it cancels one component and its negation from two clauses. One might be tempted to apply this rule to cancel more than one component from two clauses in a sigle step. We will show that this is not possible, because it would make the calculus unsound.

Example 1.120 Consider the set of clauses

$$F = \{\{x, y\},\ \{\neg x, \neg y\}\}. \tag{1.203}$$

If we could resolve more than one component at a time, then resolving simultaneously on x and y we would immediately derive the empty clause from F. However, observe that F is satisfiable: if we take any $a \in \Omega^V$ with $a(x) = 1$ and $a(y) = 0$, then $asatisfiesF$, although clearly $a \nvDash F \cup \{\bot\}$. This means that generalizing the resolution rule would make the calculs unsound, which is not what we want.

□ **Completeness.** We prove completeness by showing that the resolution inference is regular. This requires a number of steps.

Definition 1.121. If $\varphi = \{\psi_1, \dots, \psi_n\}$ is a generalized clause and ψ is any fomula, we say that a generalized clause φ' is a *ψ-variant* of φ if $\varphi' = \varphi$ or

$$\varphi' := \varphi \cup \{\psi\} = \{\psi_1, \dots, \psi_n, \psi\}\,. \tag{1.204}$$

We write $\varphi' = \varphi * \psi$ to indicate that φ' is a ψ-variant of φ. Tf $F, F' \subseteq L$ are sets of formulas and every formula of F' is a ψ-variant of a formula of F, we say that F' is a ψ-variant of F and write $F' = F * \psi$.

Note that if $\psi \in \varphi$ then $\varphi * \psi = \varphi$. Let us say that a set of formulas F expands to a formula φ if there exists an expansion of F ending in φ. Our goal here is to show that if F expands to φ, then if we add a component ψ to some clauses of F we can still expand the resulting set not necessarily to φ, but certainly to $\varphi * \psi$, that is either to φ or to $\varphi \cup \{\psi\}$. Here is the technical formulation.

Lemma 1.122. If a set of formulas $F \subseteq L$ expands to φ, then $F * \psi$ expands to $\varphi * \psi$.

Proof. We will actually prove something more detailed, namely that if $E = (\varphi_1, \dots, \varphi_n)$ is an expansion of F ending in φ, then there is an expansion of $F * \psi$ of the form $E' := (\varphi_1', \dots, \varphi_n')$ where $\varphi_k' = \varphi_k * \psi$ and such that φ_k' is obtained by applying the same expansion rule of φ_k.
Observe first that if σ is any substitution involving only the components of a clause φ and if $\varphi\sigma = \varphi'$, then

$$(\varphi * \psi)\,\sigma = \begin{cases} \varphi\sigma & \text{if } \psi \in \varphi \\ \varphi\sigma \cup \{\psi\} & \text{if } \psi \notin \varphi \end{cases} = \begin{cases} \varphi' & \text{if } \psi \in \varphi, \\ \varphi' \cup \{\psi\} & \text{if } \psi \notin \varphi \end{cases} = \varphi' * \psi. \tag{1.205}$$

The proof is now by induction: suppose the claim has been proved for all indices less than k; we prove the statement for φ_k depending on the expansion rule used to construct φ_k.

- Assumption. If $\varphi_k \in F$, then we can take $\varphi_k' = \varphi_k * \psi \in F * \psi$.
- Axiom. If φ_k is an axiom, then take $\varphi_k' = \varphi_k$.
- If φ_k is obtained from φ_i with $i < j$ using falsehood, double negation α or β-expansion, then $\varphi_k = \varphi_i\sigma$ for a suitable substitution σ involving only components of φ_i. If we set $\varphi_k' = \varphi_i'\sigma$ we have

$$\varphi_k' = \varphi_i'\sigma = (\varphi_i * \psi)\,\sigma = \varphi_i\sigma * \psi = \varphi_k * \sigma. \tag{1.206}$$

- If $\varphi_k = \varphi_i[\varnothing/\chi] \cup \varphi_j[\varnothing/\neg\chi]$ is obtained by resolution for some indices $i, j < k$, then setting $\sigma = [\varnothing/\chi]$ and $\tau = [\varnothing/\neg\chi]$, we have

$$\begin{aligned} \varphi_k' &= \varphi_i'\sigma \cup \varphi_j'\tau = (\varphi_i * \psi)\,\sigma \cup (\varphi_j * \psi)\,\tau = (\varphi_i\sigma * \psi) \cup (\varphi_j\tau * \psi) && (1.207) \\ &= (\varphi_i\sigma \cup \varphi_j\tau) * \psi = \varphi_k * \psi. && (1.208) \end{aligned}$$

Proposition 1.123. If both $F \cup \{\varphi\}$ and $F \cup \{\neg\varphi\}$ have closed expansions, then F has a closed expansion.

Proof. In $F \cup \{\neg\varphi\}$, replace the generalized clause $\{\neg\varphi\}$ with its φ-variant $\{\varphi, \neg\varphi\}$. Since $F \cup \{\neg\varphi\}$ has a closed expansion, $F \cup \{\{\varphi, \neg\varphi\}\}$ has an expansion containing either $\bot$ or its φ-variant $\{\bot, \varphi\}$ (lemma 1.122) and therefore φ by $\bot$-expansion. Since $\{\varphi, \neg\varphi\}$ s an axiom of the calculus and can always be added to an expansion of F, this means that either $\bot$ of φ belong to an expansion of F. Since $F \cup \{\varphi\}$ has a closed expansion, $\bot$ is in an expansion of F. □

Proposition 1.124. The resolution inference is an inference relation.

Proof. Assume $F, G \subseteq L$ are sets of formulas. Observe, as a preliminary remark, that if $F \subseteq G$ then every expansion of F is also an expansion of G, because the only expansion rule that depends on the set of premises is the assumption rule and if this rule is applied to a formula of F, we can regard it as applied to a formula of G.

- ○ Compactness. This amounts to prove that F has a closed expansion if and only if it has a finite subset E which has a closed expansion. However, every expansion of E is also an expansion of F. And conversely, every expansion of F has finite length, hence uses the assumption rule only finitely many times, hence is an expansion of a finite subset $E \subseteq F$.
- ○ Assumption. If $\varphi \in F$, then $F \cup \{\neg\varphi\}$ has a closed expansion as shown in (1.209) below, hence $F \vdash \varphi$.

STEP	CLAUSE	RULE
1	$\{\varphi\}$	assumption
2	$\{\neg\varphi\}$	assumption
3	$\varnothing$	1, 2, resolution

(1.209)

- ○ Consistency. If $F \vdash \varphi$ then $F \cup \{\neg\varphi\}$ has a closed expansion S, which is also a closed expansion for the larger set $F \cup \{\neg\varphi, \neg\bot\}$; this means that$F, \neg\varphi \vdash \bot$. Conversely, if $F, \neg\varphi \vdash \bot$ then $F \cup \{\neg\varphi, \neg\bot\}$ has a closed expansion S. However, S is also a closed expansion of $F \cup \{\neg\varphi\}$ because $\{\neg\bot\}$ is an axiom of the calculus. Thus $F \vdash \varphi$.
- ○ Cut. Observe that F has a closed expansion if and only if $F \vdash \bot$. In fact, $F \vdash \bot$ if and only if $F \cup \{\neg\bot\}$ has a closed expansion; however, $\neg\bot$ is one of axioms of the calculus; thus, the last condition is equivalent to the assertion that F has a closed expansion. In view of this and by proposition 1.68, to prove the cur rule it suffices to show that if both $F \cup \{\varphi\}$ and $F \cup \{\neg\varphi\}$ have closed expansions, so does F; and this is precisely the content of proposition 1.123.

PROPOSITION 1.125. Inference resolution is regular.

Proof. We use proposition 1.90. The conditions for α and β-formulas are proved below.

STEP	CLAUSE	RULE
1	$\{\alpha\}$	assumption
2	$\{\neg\alpha_i\}$	assumption
3	$\{\alpha_i\}$	1, α-expansion
4	$\varnothing$	2, 3, resolution

STEP	CLAUSE	RULE
1	$\{\beta\}$	assumption
2	$\{\neg\beta_1\}$	assumption
3	$\{\neg\beta_2\}$	1, α-expansion
4	$\{\beta_1, \beta_2\}$	1, β-expansion
5	$\{\beta_2\}$	4, 2, resolution
6	$\varnothing$	5, 3, resolution

(1.210)

As to the $\top$-sule observe that $\{\neg\top\}$ has a closed expansion by the falsehood rule and therefore $\vdash \top$. □

COROLLARY 1.126. The resolution calculus is complete.

Proof. The resolution inference is a regular inference relation (proposition 1.124 and proposition 1.124) and every regular inference relation is complete (corollary 1.92). □

1.7 The Hilbert calculus

The Hilbert calculus defines a set of axioms, a set of inference rules and the notion of derivation, from which an inference relation is defined.

□ **The induced inference relation.** We fix a propositional language L generated by the set of connectives $C = \{\bot, \top, \neg, \wedge, \vee, \to\}$; we regard biimplication as defined by the conjunction of two implications.

Definition 1.127. The *axioms* of the Hilbert calculus are all the formulas of the following types, in which α indicates a formula of conjunctive type and β one of disjunctive type.

$$\bot \to \varphi \tag{1.211}$$
$$\top \tag{1.212}$$
$$\varphi \to (\neg\varphi \to \bot) \tag{1.213}$$
$$(\varphi \to \bot) \to \neg\varphi \tag{1.214}$$
$$\neg\neg\varphi \to \varphi \tag{1.215}$$
$$\alpha \to \alpha_i \tag{1.216}$$
$$\alpha_1 \to (\alpha_2 \to \alpha) \tag{1.217}$$
$$\beta_i \to \beta \tag{1.218}$$
$$(\beta_1 \to \varphi) \to ((\beta_2 \to \varphi) \to (\beta \to \varphi)) \tag{1.219}$$
$$(\varphi \to (\chi \to \psi)) \to ((\varphi \to \chi) \to (\varphi \to \psi)) \tag{1.220}$$

Observe that all axions are valid formulas.

Definition 1.128. The only primitive *inference rule of the Hilbert calculus* is called *modus ponens* and is represented by the diagram

$$\frac{\varphi, \quad \varphi \to \psi}{\psi}. \tag{1.221}$$

We will see later that the abundance of axioms actually allows the definition of *derived* inference rules.

Definition 1.129. A *derivation* in the Hilbert calculus of a formula φ from a set of formulas $F \subseteq L$ is a finite sequence of formulas $S = (\varphi_1, \dots, \varphi_n)$ such that $\varphi_n = \varphi$ and such that every φ_i satisfies one of the following conditions.

- $\varphi_i \in F$
- φ_i is an axiom of the calculus.
- φ_i is obtained from formulas preceding it in the sequence by an application of an inference rule.

F is the set of *premises* and φ is the conclusion. A formula φ is a *theorem* of the calculus if it can be derived from the empty set of premises.

Since modus ponens is the only primitive inference rule of the Hilbert calculus, the third condition above means that there exist formulas φ_j and φ_k in the sequence with $j, k < i$ sucht that $\varphi_k = \varphi_j \to \varphi_i$.

Example 1.130 We prove that $\varphi \wedge (\varphi \to \psi) \to \psi$ is a theorem of the Hilbert calculus.

STEP	CLAUSE	RULE
1	$\varphi \wedge (\varphi \to \psi) \to \varphi$	axiom (1.216)
2	$\varphi \wedge (\varphi \to \psi) \to (\varphi \to \psi)$	axiom (1.216)
3	$(\varphi \wedge (\varphi \to \psi) \to (\varphi \to \psi)) \to$ $((\varphi \wedge (\varphi \to \psi) \to \varphi) \to (\varphi \wedge (\varphi \to \psi) \to \psi))$	axiom (1.220)
4	$(\varphi \wedge (\varphi \to \psi) \to \varphi) \to (\varphi \wedge (\varphi \to \psi) \to \psi)$	3, 2, MP
5	$\varphi \wedge (\varphi \to \psi) \to \psi$	4, 1, MP

(1.222)

Example 1.131 Here we spell out some simple consequences of the axioms.

- From axiom (1.218) taking $\beta = \psi \to \varphi$ and $i = 2$ we have
$$\varphi \to (\psi \to \varphi)\,. \tag{1.223}$$
- A special case of (1.223) is $\top \to (\varphi \to \top)$; from this, axiom (1.212) and modus ponens we derive
$$\varphi \to \top\,. \tag{1.224}$$
- Again from axiom (1.218) with $\beta = \varphi \to \psi$ with $i = 2$, this time, we have
$$\neg\varphi \to (\varphi \to \psi) \tag{1.225}$$

☐ **Soundness.** We provide a direct proof of the fact that the calculus is sound.

THEOREM 1.132. The Hilbert calculus is sound: if φ can be derived from F in the calculus, the φ is a consequence of F.
$$F \vdash \varphi \;\Rightarrow\; F \vDash \varphi \tag{1.226}$$

Proof. Observe first that all axioms of the calculus are valid formulas, a fact that can be verified with a truth table and the substitution lemma, and therefore are satisfied by any valuation. Also modus ponens, which is the only primitive inference rule, is sound, in the sense that if a valuation satisfies the premises, then it also satisfies the conclusion: if $a \vDash \varphi \to \psi$ and $a \vDash \varphi$ then from the truth table of implication we have $a \vDash \psi$.
Suppose now that $F \vdash \varphi$ and that $D = (\varphi_1, \dots, \varphi_n)$ is a derivation of φ form F. We will prove that if $a \vDash F$ then $a \vDash \varphi_k$ for every index k by induction. Suppose the claim has been proved for all φ_i's with $i < k$. We have the following cases:

- Assumption. If $\varphi_k \in F$, then $a \vDash \varphi_k$ because $a \vDash F$.
- Axiom. If φ_k is an axiom, then $a \vDash \varphi_k$ because all axioms are valid formulas.
- Modus ponens. If φ_k is the conclusion of an application of modus ponens to formulas φ_i and φ_j with $i, j < k$, then $a \vDash \varphi_i$ and $a \vDash \varphi_j$ by the inductive hypothesis and therefore $a \vDash \varphi_k$ by soundness of modus ponens.

Thus if $a \vDash F$ then $a \vDash \varphi_n = \varphi$ and therefore $F \vDash \varphi$. ☐

☐ **Completeness.** Although modus ponens is the only primitive inference rule, there is a number of derived inference rules which can be used to simplify proofs and are directly derived from the axioms.

PROPOSITION 1.133. The following are derived inference rules for the Hilbert calculus.

$$\frac{\bot}{\varphi} \quad (1.227) \qquad \frac{\neg\neg\varphi}{\varphi} \quad (1.231)$$

$$\frac{\varphi}{\top} \quad (1.228) \qquad \frac{\alpha}{\alpha_i} \quad (1.232) \qquad \frac{\beta_1 \to \varphi \quad \beta_2 \to \varphi}{\beta \to \varphi} \quad (1.235)$$

$$\frac{\varphi \quad \neg\varphi}{\bot} \quad (1.229) \qquad \frac{\alpha_1 \quad \alpha_2}{\alpha} \quad (1.233) \qquad \frac{\varphi \to (\chi \to \psi) \quad \varphi \to \chi}{\varphi \to \psi} \quad (1.236)$$

$$\frac{\varphi \to \bot}{\neg\varphi} \quad (1.230) \qquad \frac{\beta_i}{\beta} \quad (1.234)$$

Proof. As an example, we prove (1.229).

STEP	FORMULA	RULE
1	φ	assumption
2	$\neg\varphi$	assumption
3	$\varphi \to (\neg\varphi \to \bot)$	axiom (1.213)
4	$\neg\varphi \to \bot$	1, 3, MP
5	$\bot$	2, 4, MP

(1.237)

Inference rules allow to derive certain inferences from others.

PROPOSITION 1.134. Given an inference rule

$$\frac{\varphi_1 \quad \varphi_2}{\varphi} \tag{1.238}$$

and a set of formulas $F \subseteq L$, then from inferences $F \vdash \varphi_i$ for $i = 1, 2$ we can infer $F \vdash \varphi$.

Proof. Since $F \vdash \varphi_1$ there is a derivation $D_1 = (\psi_1, \dots, \psi_m)$ of $\varphi_1 = \psi_m$ from F and since $F \vdash \varphi_2$ there is a second derivation $D_2 = (\psi_{m+1}, \dots, \psi_n)$ of $\psi_n = \varphi_2$ from F. Then

$$D = (\psi_1, \dots, \psi_m, \psi_{m+1}, \dots, \psi_n, \varphi) \tag{1.239}$$

in which the last step is the inference rule (1.238) is a derivation of φ from F and therefore $F \vdash \varphi$. □

LEMMA 1.135. The following inference rules, transitivity and reflexivity, hold in the Hilbert calculus.

$$\frac{\varphi \to \chi \qquad \chi \to \psi}{\varphi \to \psi} \qquad\qquad \frac{—}{\varphi \to \varphi} \tag{1.240}$$

Proof. Transitivity is proved in the derivation below

	FORMULA	RULE
1	$\varphi \to \chi$	assumption
2	$\chi \to \psi$	assumption
3	$\varphi \to (\chi \to \psi)$	2, β-axiom
4	$(\varphi \to (\chi \to \psi)) \to$	
	$((\varphi \to \chi) \to (\varphi \to \psi))$	axiom (1.220)
5	$(\varphi \to \chi) \to (\varphi \to \psi)$	3, 4, MP
6	$\varphi \to \psi$	1, 5, MP

	FORMULA	RULE
1	$\varphi \to (\neg\varphi \to \bot)$	axiom
2	$(\neg\varphi \to \bot) \to \neg\neg\varphi$	axiom
3	$\neg\neg\varphi \to \varphi$	axiom
4	$\varphi \to \neg\neg\varphi$	1, 2, transitivity
5	$\varphi \to \varphi$	4, 5, transitivity

(1.241)

So far we have seen how to convert axioms into derived inference rule. The following tool will allow us to reverse the procedure and obtain theorems from inference rules.

Theorem 1.136. (Deduction theorem) The Hilbert inference satisfies

$$F \vdash \varphi \to \chi \;\Leftrightarrow\; F, \varphi \vdash \chi. \qquad (1.242)$$

Proof. ① Necessity is proved by the inferences on the left below; note that when we cite modus ponens we are actually using the induced inference derivation of proposition 1.134.

STEP	CLAIM	REASON
1	$F \vdash \varphi \to \chi$	hypothesis
2	$F, \varphi \vdash \varphi \to \chi$	1, weakening
3	$F, \varphi \vdash \varphi$	assumption
4	$F, \varphi \vdash \chi$	2, 3, modus ponens

STEP	CLAIM	REASON
1	$F \vdash \chi_n$	assumption
2	$F \vdash \chi_n \to (\varphi \to \chi_n)$	axiom (1.218)
3	$F \vdash \varphi \to \chi_n$	1, 2, MP

(1.243)

② Sufficiency. We prove that from a derivation S of $F, \varphi \vdash \chi$ we can costruct a derivation T of $F \vdash \varphi \to \chi$ by induction on the length $|S|$ of S. Assume the statement has been proved when $|S| < n$ and assume $S = (\chi_1, \dots, \chi_n)$ with $\chi_n = \chi$. We distinguish 3 cases depending on the rule used to obtain χ_n.

- Assumption. If $\chi_n = \varphi \in F \cup \{\varphi\}$, then either $\chi_n = \varphi$, in which case $F \vdash \varphi \to \varphi$ by lemma 1.135, the theorem rule and weakening; or $\chi_n \in F$, in which case one can use the proof on the right above.
- Axiom. If χ_n is an axiom, one uses the same proof given in the second case of assumption, except that the rule at step 1 is the axiom rule rather than assumption.
- Modus ponens. If χ_n is derived from an application of modus ponens, then $S = (\chi_1, \dots, \chi_n)$ where $i, j < n$ and $\chi_j = \chi_i \to \chi_n$. In this case we can use the inductive hypothesis at stage i and j to convert a derivation of $F, \varphi \vdash \chi_p$ into one of $F \vdash \varphi \to \chi_p$ and therefore have have

STEP	STATEMENT	REASON
1.	$F, \varphi \vdash \chi_i$	hypothesis
2.	$F, \varphi \vdash \chi_i \to \chi_n$	hypothesis
3.	$F \vdash \varphi \to \chi_i$	1, induction
4.	$F \vdash \varphi \to (\chi_i \to \chi_n)$	2, induction
5.	$F \vdash (\varphi \to (\chi_i \to \chi_n)) \to ((\varphi \to \chi_i) \to (\varphi \to \chi_n))$	axiom (1.220)
6.	$F \vdash (\varphi \to \chi_i) \to (\varphi \to \chi_n)$	5, 4, MP
7.	$F \vdash \varphi \to \chi_n$	6, 3, MP

(1.244)

The importance of the deduction theorem stems from the fact that inference rules can be translated back into theorems.

Example 1.137 Fron the transitivity inference rule (lemma 1.135) one can prove the theorem

$$(\varphi \to \chi) \to ((\chi \to \psi) \to (\varphi \to \psi)). \tag{1.245}$$

This is shown as follows:

STEP	CLAIM	REASON
1	$\varphi \to \chi,\ \chi \to \psi \vdash \varphi \to \chi$	transitivity rule
2	$\varphi \to \chi \vdash (\chi \to \psi) \to (\varphi \to \chi)$	1, DT
3	$\vdash (\varphi \to \chi) \to ((\chi \to \psi) \to (\varphi \to \chi))$	2, DT

(1.246)

Note that since the order of the elements in the set of premises is irrelevant, we can exchange the order of application of the deduction theorem on the premises and obtain as another theorem

$$(\chi \to \psi) \to ((\varphi \to \chi) \to (\varphi \to \psi)). \tag{1.247}$$

This commutativity is true for all inference rules with more than one premise.

Proposition 1.138. The Hilbert calculus defines an inference relation

Proof.

- Assumption. If $\varphi \in F$ then we have a derivation $S = (\varphi)$ whose only step uses the assumption rule.
- Compactness. If $F \vdash \varphi$, there is a finite derivation $S = (\varphi_1, \dots, \varphi_n)$ of φ and the assumption rule can only be used finitely many times. Thus, $E \vdash \varphi$ with a finite $E \subseteq F$. The converse is clear, for a derivation from E is also a derivation from a larger set F.
- Cut. Assume $F \vdash \varphi$ and $F, \varphi \vdash \psi$. By the deduction theorem, $F \vdash \varphi \to \psi$ and by modus ponens $F \vdash \psi$.
- Consistency. Necessity. If $F \vdash \varphi$, then $F, \neg\varphi \vdash \bot$ as shown on the left below. For sufficiency, use the table on the right

STEP	CLAIM	REASON
1	$F \vdash \varphi$	hypothesis
2	$F, \neg\varphi \vdash \varphi$	1, weakining
3	$F, \neg\varphi \vdash \neg\varphi$	assumption
4	$F, \neg\varphi \vdash \bot$	2, 3, (1.229)

STEP	CLAIM	REASON
1	$F, \neg\varphi \vdash \bot$	hypothesis
2	$F \vdash \neg\varphi \to \bot$	1, DT
3	$F \vdash \neg\neg\varphi$	2, (1.230)
4	$F \vdash \varphi$	3, (1.231)

(1.248)

Proposition 1.139. The Hilbert inference is a regular inference.

Proof. The rule for α-formulas follows from (1.216) and proposition 1.134: if $F \vdash \alpha$ then $F \vdash \alpha_i$. The β-rule is proved as follows.

STEP	CLAIM	REASON
1	$F, \beta_1 \vdash \bot$	hypothesis
2	$F, \beta_2 \vdash \bot$	hypothesis
3	$F \vdash \beta_1 \to \bot$	1, DT
4	$F \vdash \beta_2 \to \bot$	2, DT
5	$F \vdash \beta \to \bot$	3, 4, β-rule
6	$F, \beta \vdash \bot$	3, 4, β-rule

(1.249)

Truth follows immediately from axiom (1.212) and the weakening rule. □

Corollary 1.140. The Hilbert calculus is complete.

Proof. The Hilbert calculus induces an inference relation (proposition 1.138) which is regular (proposition 1.138) and hence complete (corollary 1.92). □

1.8 References

Smullyan [1]