Introduction

**STRUCTURAL PROOF THEORY**

The idea of mathematical proof is very old, even if precise principles of proof have been laid down during only the past hundred years or so. Proof theory was first based on axiomatic systems with just one or two rules of inference. Such systems can be useful as formal representations of what is provable, but the actual finding of proofs in axiomatic systems is next to impossible. A proof begins with instances of the axioms, but there is no systematic way of finding out what these instances should be. Axiomatic proof theory was initiated by David Hilbert, whose aim was to use it in the study of the consistency, mutual independence, and completeness of axiomatic systems of mathematics.

**Structural proof theory** studies the general structure and properties of mathematical proofs. It was discovered by Gerhard Gentzen (1909–1945) in the first years of the 1930s and presented in his doctoral thesis *Untersuchungen über das logische Schliessen* in 1933. In his thesis, Gentzen gives the two main formulations of systems of logical rules, **natural deduction** and **sequent calculus**. The first aims at a close correspondence with the way theorems are proved in practice; the latter was the formulation through which Gentzen found his main result, often referred to as Gentzen’s “Hauptsatz.” It says that proofs can be transformed into a certain “cut-free” form, and from this form general conclusions about proofs can be made, such as the consistency of the system of rules.

The years when Gentzen began his researches were marked by one great but puzzling discovery, Gödel’s incompleteness theorem for arithmetic in 1931: Known principles of proof are not sufficient for deriving all of arithmetic; moreover, no single system of axioms and rules can be sufficient. Gentzen’s studies of the proof theory of arithmetic led to **ordinal proof theory**, the general task of which is to study the deductive strength of formal systems containing infinitistic principles of proof. This is a part of proof theory we shall not discuss.

Of the two forms of structural proof theory that Gentzen gave in his doctoral thesis, natural deduction has remained remarkably stable in its treatment of rules.
of proof. Sequent calculus, instead, has been developed in various directions. One line leads from Gentzen through Ketonen, Kleene, Dragalin, and Troelstra to what are known as *contraction-free* systems of sequent calculus. Each of these logicians added some essential discovery, until a gem emerged. What it is can be only intimated at this stage: There is a way of organizing the principles of proof so that one can start from the theorem to be proved, then analyze it into simpler parts in a guided way. The gem is this “guided way”: namely, if one lays down what the last rule of inference was, the premises of that last step are uniquely determined. Next, one goes on analyzing these premises, and so on. Gentzen’s basic discovery is reformulated as stating that a proof can be so organized that the premises of each step of inference are always simpler than its conclusion. (To be more accurate, it can also happen that the premises are not more complicated than the conclusion.)

Given a purported theorem, the question is whether it is provable or unprovable. In the first case, the task is to find a proof. In the second case, the task is to show that no proof can exist. How can we, then, prove unprovability? The possibility of such proofs depends crucially on having the right kind of calculus, and these proofs can take various forms: In the simplest cases we go through all the rules and find that none of them has a conclusion of the form of the claimed theorem. For certain classes of theorems, we can show that it makes no difference in what order we analyze the theorem to be proved. Each step of analysis leads to simpler premises and therefore the process stops. From the way it stops we can decide if the conclusion really is a theorem or not. In other cases, it can happen that the premises are at least as complicated as the conclusion, and we could go on indefinitely trying to find a proof. Some ingenious discovery is usually needed to prove unprovability, say, some analyses stop without giving a proof, and we are able to show that all of the remaining alternatives never stop and thus never give a proof.

One line of division in proof theory concerns the methods used in the study of the structure of proofs. In his original proof-theoretical program, Hilbert aimed at an “absolutely reliable” proof of consistency for formalized mathematics. The methods he thought acceptable had to be finitary, but the goal was shown to be unattainable already for arithmetic by Gödel’s results. Later, parts of proof theory remained **reductive**, using different constructive principles, whereas other parts have studied proofs by unrestricted means. Most of our methods can be classified as reductive, but the reasons for restricted methods do not depend on arguments such as reliability. It is rather that we want results about systems of formal proof to have a computational significance. Thus it would not be sufficient to show by unrestricted means the mere existence of proofs with some desirable property. Instead, a constructive method for actually transforming any given proof so that it has the property is sought. From this point of view, our treatment of
structural proof theory belongs, with a few exceptions, to what can be described as **computational proof theory**.

Since 1970, a branch of proof theory known as **constructive type theory** has been developed. A theorem typically states that a certain claim holds under given assumptions. The basic idea of type theory is that proofs are **functions** that convert any proofs of the assumptions of a theorem into a proof of its claim. A connection to computer science is established: In the latter, formal languages have been developed for constructing functions (programs) that act in a specified way on their input, but there has been no formal language for expressing what this specified way, the **program specification**, is. Logical languages, in turn, are suitable for expressing such specifications, but they have totally lacked a formalism for constructing functions that effect the task expressed by the specification. Constructive type theory unites specification language and programming language in a unified formalism in which the task of verifying the correctness of a program is the **same** as the logical task of controlling the correctness of a formal proof. We do not cover constructive type theory in detail, as another book would be needed for that, but some of the basic ideas and their connection to natural deduction and normalization procedures are explained in Appendix B.

At present, there are many projects in the territory between logic, mathematics, and computer science that aim at fully formalizing mathematical proofs. These projects use computer implementations of **proof editors** for the interactive development of formal proofs, and it cannot be said what all the things are that could come out of such projects. It has been observed that even the most detailed informal proofs easily contain gaps and cannot be routinely completed into formal proofs. More importantly, one finds imprecision in the conceptual foundation. The most optimistic researchers find that formalized proofs will become the standard in mathematics some day, but experience has shown formalization beyond the obvious results to be time-consuming. At present, proof editors are still far from being practical tools for the mathematician. If they gain importance in mathematics, it will be due to a change in emphasis through the development of computer science and through the interest in the computational content of mathematical theories. On the other hand, proof editors have been used for program verification even with industrial applications for some years by now. Such applications are bound to increase through the critical importance of program correctness.

Gentzen's structural proof theory has achieved perfection in pure logic, the predicate calculus. Intuitionistic natural deduction and classical sequent calculus are by now mastered, but the extension of this mastery beyond pure logic has been limited. A new approach that we exploit is to formulate mathematical theories as systems of **nonlogical rules** added to a suitable sequent calculus. As examples of proof analyses, theories of order, lattice theory, and plane affine geometry are treated. These examples indicate a way to an emerging field of study that could
be called **proof theory in mathematics**. It is interesting to note that a large part of abstract mathematical reasoning seems to be finitary, thus not requiring any strong transfinite methods in proof analysis.

In the rest of this Introduction we comment on use of this book in teaching proof theory and what is new in it.

**USE OF THIS BOOK IN TEACHING**

Chapters 1–4 are based on courses in proof theory we have given at the University of Helsinki. The main objective of these courses was to give to the students a concise introduction to contraction-free intuitionistic and classical sequent calculi. The first author has also given a more specialized course on natural deduction, based on Chapter 1, the first two sections of Chapter 5, Chapter 8, and Appendices A and B.

The presentation is self-contained and the book should be readable without any previous knowledge of logic. Some familiarity with the topic, as in Van Dalen’s *Logic and Structure*, will make the task less demanding.

Chapter 1 starts with general observations about logical languages and rules of inference. In a first version, we had defined logical languages through categorial grammars, but this was judged too difficult by most colleagues who read the text. With some reluctance, the categorial grammar approach was moved to Appendix A. Some traces of the definition of logical languages through an abstract syntax remained in the first section of Chapter 1, though.

The introduction rules of natural deduction are explained through the computational semantics of intuitionistic logic. A generalization of the inversion principle, to the effect that “whatever follows from the direct grounds for deriving a proposition, must follow from that proposition,” determines the corresponding elimination rules. By the inversion principle, three rules, those of conjunction elimination, implication elimination, and universal elimination, obtain a form more general than the standard natural deduction rules. Using these general elimination rules, we are able to introduce sequent calculus rules as formalizations of the derivability relation in natural deduction. Contraction-free intuitionistic and classical sequent calculi are treated in detail in Chapters 2 and 3. These chapters work as a concise introduction to the central methods and general results of structural proof theory. The basic parts of structural proof theory use combinatorial reasoning and elementary induction on formula length, height of derivation, and so on, therefore perhaps giving an impression of easiness on the newcomer. The main difficulty, witnessed by the long development of structural proof theory, is to find the right rules. The first part of our text shows in what order structural proof theory is built up once those rules have been found. The second part of the
book, Chapters 5–8, gives ample further illustration of the methodology. There is usually a large number of details, and a delicate order is required for putting things together, and mistakes happen. For such reasons, our first cut elimination theorem, in Chapter 2, considers, to our knowledge, absolutely all cases, even at the expense of perhaps being a bit pedantic.

In Chapter 3, following a suggestion of Gentzen, multisuccedent sequents are presented as a natural generalization of single succedent sequents into sequents with several (classical) open cases in the succedent. We find it important for students to avoid the denotational reading of sequents in favor of one in terms of formal proofs.

Chapter 4 contains a systematic treatment of quantifier rules in sequent calculi, again introduced through natural deduction and the general inversion principle.

Connected to the book is an interactive proof editor for developing formal derivations in sequent calculi. The system has been implemented by Aarne Ranta in the functional programming language Haskell. A description of the system, with instructions on how to access and use it, is given in Appendix C written by Ranta.

The proof editor serves several purposes: First, it makes the development of formal derivations in sequent calculi less tedious, thereby helping the student. It also checks the formal correctness of derivations. The user can give axiomatic systems to the editor that converts them into systems of nonlogical rules of inference by which the logical sequent calculi are extended. Formal derivations are quite feasible to develop in those extensions we have so far studied. Even though the extensions need not permit a terminating proof search, the user will soon notice how neatly the search space can become limited, often into one or two applicable rules only, or no rules at all, which establishes underviability.

The proof editor produces provably correct \LaTeX\ code, with the advantage that the rewriting of parts of sequents is not needed. The editor is in its early stages; more is hoped to be included in later releases, including translation algorithms between various calculi, cut elimination algorithms, a natural language interface, and so on.

\textbf{Exercises}, mostly to Chapters 1–4 and 8, can be found in the book’s home page (see p. 243). We welcome suggestions for further exercises. Basic exercises are just derivations of formulas in the various calculi. Other exercises consist in filling in details of proofs of theorems. Another type of task, for those conversant with the Haskell language, is to formalize theorems about sequent calculi. Since these theorems are, almost without exception, proved constructively in the book, their formalizations give proof-theoretical algorithms for the transformation of proofs. An example is the proof of Glivenko’s theorem in Section 5.4.

Through the use of contraction-free sequent calculi, it is possible for students to find proofs of results that were published as research results only a few decades
ago, say, Harrop's theorem in Section 2.5. This should give some idea of what a powerful tool is being put into their hands.

Finally, a description of what is new in this book (for the expert, mostly).

**WHAT IS NEW IN THIS BOOK?**

Chapter 1 contains a generalization of the inversion principle of Gentzen and Prawitz, one that leads to elimination rules that are more general than the usual ones. Contrary to earlier inversion principles that only justify the elimination rules, our principle actually determines what the elimination rules corresponding to given introduction rules should be. The elimination rules are all of the form of disjunction elimination, with an arbitrary consequence.

Starting from natural deduction with general elimination rules, the rules of sequent calculus are presented in Section 1.3 as straightforward formalizations of the derivability relation of natural deduction.

Section 3.3 gives a proof of completeness of the contraction-free invertible sequent calculus for classical propositional logic known as $G3c$-calculus in the literature. The proof is an elaboration of Ketonen's original 1944 proof. It uses a novel notion of validity defined as a negative concept, the inexistence of a refuting valuation.

Chapter 4 contains proofs of height-preserving $\alpha$-conversion and the substitution lemma that we have not found done elsewhere in such detail. Section 4.4 gives a proof of completeness of classical predicate calculus worked out for the definition of validity as a negative notion.

Chapter 5 studies various sequent calculi, most of which are new to the literature. Section 5.1 gives a sequent calculus with independent contexts in all two-premiss rules and explicit rules of weakening and contraction. The calculus is motivated by the independent treatment of assumptions in natural deduction. A classical multisuccedent version is also given. Proofs of cut elimination for these calculi are given that do not use Gentzen's mix rule, or rule of multicut. In Section 5.2, the calculi of Section 5.1 are modified so that weakening and contraction are treated implicitly as in natural deduction. Section 5.4 gives a single succedent calculus for classical propositional logic based on a formulation of the law of excluded middle as a sequent calculus rule. A proof of Glivenko's theorem is given through an explicit proof transformation. It is shown that in the derivation of a sequent $\Gamma \Rightarrow C$, the rule can be restricted to atoms of $C$, from which a full subformula property follows.

Chapter 6 studies extensions of logical sequent calculi by nonlogical rules corresponding to axioms. Contrary to widespread belief, it is possible to add axioms to sequent calculus as rules of a suitable form while maintaining the eliminability of cut. These extensions have no structural rules, which gives a
strong control over the structure of possible derivations. As a first application, a formulation of predicate calculus with equality as a cut-free system of rules is given. It is proved through an explicit proof transformation that predicate logic with equality is conservative over predicate logic. It is essential for the proof that no cuts, even on atoms, be permitted. In Section 6.6, examples of structural proof analysis in mathematics are given. Topics covered include intuitionistic and classical theories of order, lattice theory, and affine geometry. The last one goes beyond the expressive means of first-order logic, but the methods of structural proof analysis of the previous chapters still apply. As an example of such analysis of derivations without structural rules, a proof of independence of Euclid’s fifth postulate in plane affine geometry is given.

In Chapter 8 the structure of derivations in natural deduction with general elimination rules is studied. A uniform definition of normality of derivations is given: A derivation is normal when all major premisses of elimination rules are assumptions. This structure follows from the applicability of permutation conversions to all cases in which the major premiss of an elimination rule is concluded by another elimination rule. Translations are given that establish isomorphism of normal derivations and cut-free derivations in the sequent calculus with independent contexts of Chapter 5. It is shown what the role of the structural rules of sequent calculus is in terms of natural deduction: Weakening and contraction in sequent calculus correspond to the vacuous and multiple discharge of assumptions in natural deduction. Cuts in which the cut formula is principal in the right premiss correspond to such steps of elimination in which the major premiss has been derived. (No other cuts can be expressed in terms of natural deduction.) A translation from non-normal derivations to derivations with cuts is given, from which follows a normalization procedure consisting of said translation, followed by cut elimination and translation back to natural deduction.

In the Conclusion, a uniform logical calculus is given that encompasses both sequent calculus and natural deduction.
We first discuss logical languages and rules of inference in general. Then the rules of natural deduction are presented, with introduction rules motivated by meaning explanations and elimination rules determined by an inversion principle. A way is found from the rules of natural deduction to those of sequent calculus. In the last section, we discuss some of the main characteristics of structural proof analysis in sequent calculus.

1.1. LOGICAL SYSTEMS

A logical system consists of a formal language and a system of axioms and rules for making logical inferences.

(a) Logical languages: A logical language is usually defined through a set of inductive clauses for well-formed formulas. The idea is that expressions of a formal language are special sequences of symbols from a given alphabet, as generated by the inductive definition. An alternative way of defining formal languages is through categorical grammars. Such grammars are well-known for natural languages, and categorical grammars for formal languages are in use with programming languages, but not so often in logic.

Under the first approach, expressions of a logical language are formulas defined inductively by two clauses: 1. A statement of what the prime formulas are. These are formulas that contain no other formulas as parts. 2. A statement of what the compound formulas are. These are built from other simpler formulas by logical connectives, and their definition requires reference to arbitrary formulas and how these can be put together with the symbols for connectives to give new formulas. Given a formula, we can find out how it was put together from other formulas and a logical connective. Parentheses may be needed to indicate the composition uniquely. Then we can find out how the parts were obtained until we arrive at the prime formulas. Thus, in the end, all formulas consist of prime formulas, logical connectives, and parentheses.
We shall define the language of **propositional logic**:

1. The prime formulas are the **atomic** formulas denoted by $P, Q, R, \ldots$, and **falsity** denoted by $\bot$.
2. If $A$ and $B$ are formulas, then the **conjunction** $A \& B$, **disjunction** $A \vee B$, and **implication** $A \supset B$ are formulas.

For unique readability of formulas, the components should always be put in parentheses but in practice these are left out if a conjunction or disjunction is a component of an implication. Often $\bot$ is counted among the atomic formulas, but this will not work in proof theory. It is best viewed as a zero-place connective. **Negation** $\sim A$ and **equivalence** $A \supset \subseteq B$ are defined as $\sim A = A \supset \bot$ and $A \supset \subseteq B = (A \supset B) \& (B \supset A)$.

Expressions of a language should express something, not just be strings of symbols from an alphabet put together correctly. In logic, the thing expressed is called a **proposition**. Often, instead of saying “proposition expressed by formula $A$” one says simply “proposition $A$.” There is a long-standing debate in philosophy on what exactly propositions are. When emphasis is on logic, and not on what logic in the end of a philosophical analysis is, one considers expressions in the formal sense and talks about formulas.

In recent literature, the definition of expressions as sequences of symbols is referred to as **concrete syntax**. Often it is useful to look at expressions from another point of view, that of **abstract syntax**, as in categorial grammar. The basic idea of categorial grammar is that expressions of a language have a **functional structure**. For example, the English sentence *John walks* is obtained by representing the intransitive verb *walk* as a function from the category of noun phrases $NP$ to the category of sentences $S$, in the usual notation for functions, *walk* : $NP \rightarrow S$. *John* is an element of the category $NP$ and **application** of the function *walk* gives as value *walk*(*John*), an element of the category of sentences $S$. One further step of **linearization** is required for hiding the functional structure, to yield the original sentence *John walks*. In logic and mathematics, no consideration is given to differences produced by this last stage, nor to differences in the grammatical construction of sentences. Since Frege, one considers only the logical content of the functional structure.

We shall briefly look at the definition of propositional logic through a categorial grammar. There is a **basic category** of propositions, designated $Prop$. The atomic propositions are introduced as parameters $P, Q, R, \ldots$, with no structure and with the categorizations

$$P : Prop, \quad Q : Prop, \quad R : Prop, \ldots$$

and similarly for falsity, $\bot : Prop$. The connectives are two-place functions for forming new propositions out of given ones. Application of the function $\&$ to the
two arguments $A$ and $B$ gives the proposition $\& (A, B)$ as value, and similarly for $\lor$ and $\Rightarrow$. The functional structure is usually hidden by an infix notation and by the dropping of parentheses, $A \& B$ for $\& (A, B)$, and so on. This will create an ambiguity not present in the purely functional notation, such as $A \& B \Rightarrow C$ that could be both $\& (A, \Rightarrow (B, C))$ and $\Rightarrow (\& (A, B), C)$. As mentioned, we follow the convention of writing $A \& (B \Rightarrow C)$ for the former and $A \& B \Rightarrow C$ for the latter, and in general, having conjunction and disjunction bind more strongly than implication. Appendix A explains in more detail how logical languages are treated from the point of view of categorial grammar.

Neither approach, inductive definition of strings of symbols nor generation of expressions through functional application, reveals what is special about logical languages. Logical languages of the present day have arisen as an abstraction from the informal language of mathematics. The first work in this direction was by Frege, who invented the language of predicate logic. It was meant to be, wrote Frege, "a formula language for pure thought, modeled upon that of arithmetic." Later Peano and Russell developed the symbolism further, with the aim of formalizing the language of mathematics. These pioneers of logic tried to give definitions of what logic is, how it differs from mathematics, and whether the latter is reducible to the former, or if it is perhaps the other way around.

From a practical point of view there is a clear understanding of what logical languages are: The prime logical languages are those of propositional and predicate logic. Then there are lots of other logical languages more or less related to these. Logic itself is, from this point of view, what logicians study and develop. Any general definition of logic and logical languages should respect this situation.

An essential aspect of logical languages is that they are formal languages, or can easily be made into such, an aspect made all the more important by the development of computer science. There are many connections between logical languages and programming languages; in fact, logical and programming languages are brought together in one language in some recent developments, as explained in Appendix B.

(b) Rules of inference: Rules of inference are of the form: "If it is the case that $A$ and $B$, then it is the case that $C$." Thus they do not act on propositions, but on assertions. We obtain an assertion from a proposition $A$ by adding something to it, namely, an assertive mood such as "it is the case that $A". Frege used the assertion sign $\vdash A$ to indicate this but usually the distinction between propositions and assertions is left implicit. Rules seemingly move from given propositions to new ones.

In Hilbert-style systems, also called axiomatic systems, we have a number of basic forms of assertion, such as $\vdash A \Rightarrow A \lor B$ or $\vdash A \Rightarrow (B \lor A)$. Each instance of these forms can be asserted, and in the case of propositional logic there
is just one rule of inference, of the form

\[ \vdash A \supset B \vdash A \]

\[ \vdash B \]

Derivations start with instances of axioms that are decomposed by the rule until the desired conclusion is found.

In natural deduction systems, there are only rules of inference, plus assumptions to get derivations started, exemplified by

\[ [\vdash A] \]

\[ \vdots \]

\[ \vdash A \vdash B \]

\[ \vdash A \supset B \]

\[ \vdash B \]

\[ \vdash A \supset B \]

Instances of the first rule are single-step inferences, and if the premisses have been derived from some assumptions, the conclusion depends on the same assumptions. In the second rule instead, in which the vertical dots indicate a derivation of \( \vdash B \) from \( \vdash A \), the assumption \( \vdash A \) is discharged at the inference line, as indicated by the square brackets, so that \( \vdash B \) above the inference line depends on \( \vdash A \) whereas \( \vdash A \supset B \) below it does not.

In sequent calculus systems, there are no temporary assumptions that would be discharged, but an explicit listing of the assumptions on which the derived assertion depends. The derivability relation, to which reference was made in natural deduction by the four vertical dots, is an explicit part of the formal language, and sequent calculus can be seen as a formal theory of the derivability relation.

Of the three types of systems, the first, axiomatic, has some good properties that are due to the presence of only one rule of inference. However, it is next to impossible to actually use the axiomatic approach because of the difficulty of finding the instances of axioms to start with. Systems of the second type correspond to the usual way of making inferences in mathematics, with a good sense of structure. Systems of the third type are the ones that permit the most profound analysis of the structure of proofs, but their actual use requires some practice. Moreover, the following is possible in natural deduction and in sequent calculus:

\[ \text{Two systems of rules can be equivalent in the sense that the same assertions can be derived in them, but the addition of the same rule to each system can destroy the equivalence.} \]

This lack of modularity will not occur with the axiomatic Hilbert-style systems.

Once a system of rules of logical inference has been put up, it can be considered from the formal point of view. The assertion sign is left out, and rules of inference are just ways of writing a formula under any formula or formulas that have the form of the premisses of the rules. In a complete formalization of logic, also the
formation of propositions is presented as the application of rules of proposition formation. For example, conjunction formation is application of the rule

\[
\frac{A : Prop \quad B : Prop}{A \& B : Prop}
\]

Rules of inference can be formalized in the same way as rules of proposition formation: They are represented as functions that take as arguments formal proofs of the premisses and give as value a formal proof of the conclusion. A hierarchy of functional categories is obtained such that all instances of rules of proposition formation and of inference come out through functional application. This will lead to [constructive type theory](#), described in more detail in Appendix B.

The viewpoint of proof theory is that logic is the theory of correct demonstrative inference. Inferences are analyzed into the most basic steps, the formal correctness of which can be easily controlled. Moreover, the semantical justification of inferences can be made compositional through the justification of the individual steps of inference and how they are put together.

Compound inferences are synthesized by the composition of basic steps of inference. A system of rules of inference is used to give an inductive, formal definition of the notion of [derivation](#). Derivability then means the existence of a derivation. The correctness of a given derivation can be mechanically controlled through its inductive definition, but the finding of derivations typically is a different matter.

### 1.2. Natural Deduction

Natural deduction embodies the operational or computational meaning of the logical connectives and quantifiers. The meaning explanations are given in terms of the [immediate grounds](#) for asserting a proposition of corresponding form. There can be other, less direct grounds, but these should be reducible to the former for a coherent operational semantics to be possible. The “BHK-conditions” (for Brouwer–Heyting–Kolmogorov), which give the explanations of logical operations of propositional logic in terms of [direct provability](#) of propositions, can be put as follows:

1. A direct proof of the proposition \( A \& B \) consists of proofs of the propositions \( A \) and \( B \).
2. A direct proof of the proposition \( A \lor B \) consists of a proof of the proposition \( A \) or a proof of the proposition \( B \).
3. A direct proof of the proposition \( A \supset B \) consists of a proof of the proposition \( B \) from the assumption that there is a proof of the proposition \( A \).
4. A direct proof of the proposition \( \bot \) is impossible.
In the third case it is only assumed that there is a proof of \( A \), but the proof of the conclusion \( A \supset B \) does not depend on this assumption temporarily made in order to reduce the proof of \( B \) into a proof of \( A \). Proof here is an informal notion. We shall gradually replace it by the formal notion of derivability in a given system of rules.

We can now make more precise the idea that rules of inference act on assertions; namely, an assertion is warranted if there is a proof available, and therefore, on a formal level, rules of inference act on derivations of the premisses, to yield as value a derivation of the conclusion. From the BHK-explanations, we arrive at the following introduction rules:

\[
\begin{align*}
A & \quad \frac{B}{A \& B} \quad \&I \\
\vdots & \\
A & \quad \frac{A \lor B}{B} \quad \lor I_1 \\
\vdots & \\
B & \quad \frac{A \lor B}{A \supset B} \quad \lor I_2 \\
\vdots & \\
& \quad \frac{[A]}{\supset I} \\
& \\
\end{align*}
\]

The assertion signs are left out. (There will be another use for the symbol soon.) In the last rule the auxiliary assumption \( A \) is discharged at the inference, which is indicated by the square brackets. We have as a special case of implication introduction, with \( B = \bot \), an introduction rule for negation. There is no introduction rule for \( \bot \).

There will be elimination rules corresponding to the introduction rules. They have a proposition of one of the three forms, conjunction, disjunction, or implication, as a major premiss. There is a general principle that helps to find the elimination rules: We ask what the conditions are, in addition to assuming the major premiss derived, that are needed to satisfy the following:

**Inversion principle:** Whatever follows from the direct grounds for deriving a proposition must follow from that proposition.

For conjunction \( A \& B \), the direct grounds are that we have a derivation of \( A \) and a derivation of \( B \). Given that \( C \) follows when \( A \) and \( B \) are assumed, we thus find, through the inversion principle, the elimination rule

\[
\begin{align*}
A \& B & \quad \frac{C}{[A, B]} \quad \&I \\
\vdots & \\
A \& B & \quad \frac{C}{C} \quad \&E \\
\end{align*}
\]

The assumptions \( A \) and \( B \) from which \( C \) was derived are discharged at the inference. If in a derivation the premisses \( A \) and \( B \) of the introduction rule have been
derived and $C$ has been derived from $A$ and $B$, the derivation

\[
\begin{array}{c}
\vdots \\
\null \\
A \\
\hline \\
A \land B \\
\hline \\
A \land B \\
\hline \\
C \\
\hline \\
C \land E
\end{array}
\]

converts into a derivation of $C$ without the introduction and elimination rules,

\[
\begin{array}{c}
\vdots \\
\null \\
A \\
\hline \\
B \\
\hline \\
C
\end{array}
\]

Therefore, if $\& I$ is followed by $\& E$, the derivation can be simplified.

For disjunction, we have two cases. Either $A \lor B$ has been derived from $A$ and $C$ is derivable from assumption $A$, or it has been derived from $B$ and $C$ is derivable from assumption $B$. Taking into account that both cases are possible, we find the elimination rule

\[
\begin{array}{c}
\vdots \\
\null \\
A \\
\hline \\
B \\
\hline \\
A \lor B \\
\hline \\
C \\
\hline \\
C \lor E
\end{array}
\]

Assume now that $A$ or $B$ has been derived. If it is the former and if $C$ is derivable from $A$ and $C$ is derivable from $B$, the derivation

\[
\begin{array}{c}
\vdots \\
\null \\
A \\
\hline \\
A \lor B \\
\hline \\
C \\
\hline \\
C \lor E
\end{array}
\]

converts into a derivation of $C$ without the introduction and elimination rules,

\[
\begin{array}{c}
\vdots \\
\null \\
A \\
\hline \\
C
\end{array}
\]

In the latter case of $B$ having been derived, the conversion is into

\[
\begin{array}{c}
\vdots \\
\null \\
B \\
\hline \\
C
\end{array}
\]
Again, an introduction followed by the corresponding elimination can be removed from the derivation.

The elimination rule for implication is harder to find. The direct ground for deriving \( A \supset B \) is the existence of a \textit{hypothetical} derivation of \( B \) from the assumption \( A \). The fact that \( C \) can be derived from the existence of such a derivation can be expressed by:

\[
\text{If } C \text{ follows from } B, \text{ then it already follows from } A.
\]

This is achieved precisely by the elimination rule

\[
\frac{B \quad \vdots}{A \supset B \qquad A \quad C \quad \supset E}{\vdots}
\]

In addition to the major premiss \( A \supset B \), there is the \textit{minor premiss} \( A \) in the \( \supset E \) rule. If \( B \) has been derived from \( A \) and \( C \) from \( B \), the derivation

\[
\frac{[A] \quad \vdots \quad [B] \quad \vdots}{B \quad \supset I \quad \vdots}{A \quad \supset B \quad \supset E \quad \vdots}{A \quad C \quad \supset E}
\]

converts into a derivation of \( C \) from \( A \) without the introduction and elimination rules,

[diagram]

Finally we have the zero-place connective \( \bot \) that has no introduction rule. The immediate grounds for deriving \( \bot \) are empty, and we obtain as a limiting case of the inversion principle the rule of \textit{falsity elimination} ("ex falso quodlibet") that has only the major premiss \( \bot \):

\[
\frac{\bot \quad \bot E}{C}
\]

We have still to tell how to get derivations started. This is done by the \textit{rule of assumption} that permits us to begin a derivation with any formula. In a given derivation tree, those formula occurrences are assumptions, or more precisely,
open assumptions, that are neither conclusions nor discharged by any rule. Discharged assumptions are also called closed assumptions.

Rules &E and ⊃E are usually written for only the special cases of C = A and C = B for &E and C = B for ⊃E, as follows:

\[
\begin{align*}
\frac{A \& B}{A} & \quad & \frac{A \& B}{B} \\
\quad & & \frac{A \supset B}{A} & \quad & \frac{A \supset B}{A \supset E}
\end{align*}
\]

These “special elimination rules” correspond to a more limited inversion principle, one requiring that elimination rules conclude the immediate grounds for deriving a proposition instead of arbitrary consequences of these grounds. The first two rules just conclude the premises of conjunction introduction. The third gives a one-step derivation of B from A by a rule that is often referred to as “modus ponens.” The more limited inversion principle suffices for justifying the special elimination rules but is not adequate for determining what the elimination rules should be. In particular, it says nothing about ⊥E.

The special elimination rules have the property that their conclusions are immediate subformulas of their premises. With conjunction introduction, it is the other way around: The premises are immediate subformulas of the conclusion. Further, in implication introduction, the formula above the inference line is an immediate subformula of the conclusion. It can be shown that derivations with conjunction and implication introduction and the special elimination rules can be transformed into a normal form. The transformation is done by detour conversions, the removal of applications of introduction rules followed by corresponding elimination rules. In a derivation with special elimination rules in normal form, first, assumptions are made, then elimination rules are used, and finally, introduction rules are used. This simple picture of normal derivations, moving by elimination rules from assumptions to immediate subformulas and then by introductions the other way around, is lost with the disjunction elimination rule. However, when all elimination rules are formulated in the general form, a uniform subformula structure for natural deduction derivations in normal form is achieved. The normal form itself is characterized by the following property:

**Normal form:** A derivation in natural deduction with general elimination rules is in normal form if all major premises of elimination rules are assumptions.

In general, it need not be the case that a system of natural deduction admits conversion to normal form, but it is the aim of structural proof theory to find systems that do. There is a series of properties of growing strength relating to normal form, the weakest being the existence of normal form. This property states that if a formula A is derivable in a system, there exists also a derivation of A in normal form. Secondly, we have the concept of normalization: A procedure
is given for actually converting any given derivation into normal form. The next notion is strong normalization: The application of conversions to a non-normal derivation in any order whatsoever terminates after some number of steps in a derivation in normal form. Last, we have the notion of uniqueness of normal form: The process of normalization of a given derivation always terminates with the same normal proof. Note that it does not follow that there would be only one normal derivation of a formula, for different non-normal derivations would in general terminate in different normal derivations.

The conjunction and the disjunction introduction rules, as well as the special elimination rules for conjunction and implication, are simple one-step inferences. The rest of the rules are schematic, with “vertical dots” that indicate derivations with assumptions. The behavior of these assumptions is controlled by discharge functions: Each assumption gets a number and the discharge of assumptions is indicated by writing the number next to the inference line. Further, the discharge is optional, i.e., we can, and indeed sometimes must, leave an assumption open even if it could be discharged.

Some examples will illustrate the management of assumptions and point at some peculiarities of natural deduction derivations.

Example 1:

\[
\begin{array}{c}
1. [A] \\
A \supset A \quad \cup I, 1.
\end{array}
\]

The rule schemes of natural deduction display only the open assumptions that are active in the rule, but there may be any number of other assumptions. Thus the conclusion may depend on a whole set \( \Gamma \) of assumptions, which can be indicated by the notation \( \Gamma \vdash A \). Now the rule of implication introduction can be written as

\[
\frac{\Gamma \vdash B}{\Gamma - \{A\} \vdash A \supset B \quad \cup I}
\]

In words, if there is a derivation of \( B \) from the set of open assumptions \( \Gamma \), there is a derivation of \( A \supset B \) from assumptions \( \Gamma \) minus \( \{A\} \). In this formulation there is a “compulsory” discharge of the assumption \( A \). All the other rules of natural deduction can be written similarly. We give two examples:

\[
\begin{array}{c}
\Gamma \vdash A \quad \Delta \vdash B \quad \& I \\
\Gamma \cup \Delta \vdash A \& B \\
\Gamma \vdash A \lor B \quad \Delta \cup \{A\} \vdash C \quad \Theta \cup \{B\} \vdash C \quad \lor E \\
\Gamma \cup \Delta \cup \Theta \vdash C
\end{array}
\]

The resulting system of inference, introduced by Gentzen in 1936, is usually known as “natural deduction in sequent calculus style.” It can be used to clarify the strange-looking derivation of Example 1: The assumption of \( A \) is written as
A ⊨ A, and we have the derivation

\[
\begin{align*}
A & \vdash A \\
\vdash A \supset A & \vdash I
\end{align*}
\]

The first occurrence of \( A \) has the set of assumptions \( \Gamma = \{ A \} \), and so (dropping the braces around singleton sets) the conclusion has the set of assumptions \( A \setminus A = \emptyset \).

Example 2 shows how superfluous assumptions can be added to weaken the consequent \( A \) of the first example into \( B \supset A \).

Example 2:

\[
\begin{align*}
\frac{1. \quad [A]}{B \supset A} & \vdash I \\
\vdash A \supset (B \supset A) & \vdash I,1.
\end{align*}
\]

The first inference step is justified by the rule about sets of assumptions: \( A \setminus B = A \). There is a vacuous discharge of \( B \) in the first instance of \( \vdash I \), and discharge of \( A \) takes place only at the second instance of \( \vdash I \). Note that there is a problem here in the case of \( B = A \), for compulsory discharge dictates that \( A \) is discharged at the first inference, the second becoming a vacuous discharge. The instance of the derivation with \( B = A \) is not a syntactically correct one; therefore the original derivation cannot be correct either. Chapter 8 will give a method for handling the discharge of assumptions, the unique discharge principle, that does not lead to such problems.

In sequent calculus style, the derivation is

\[
\begin{align*}
\frac{A \vdash A}{A \vdash B \supset A} & \vdash I \\
\vdash A \supset (B \supset A) & \vdash I
\end{align*}
\]

Example 3 gives a derivation that cannot be done with just a single use of assumption \( A \).

Example 3:

\[
\begin{align*}
\frac{2. \quad [A \supset (A \supset B)] \quad 1. \quad [A]}{A \supset B} & \supset E \\
\vdash [A] & \vdash E \\
\frac{B}{A \supset B} & \vdash I,1.
\end{align*}
\]

\[
\begin{align*}
\frac{A \supset B}{(A \supset (A \supset B)) \supset (A \supset B)} & \vdash I,2.
\end{align*}
\]

Assumption \( A \) had to be made twice, and there is correspondingly a multiple discharge at the first instance of \( \vdash I \) with both occurrences of assumption \( A \) discharged. Note the “nonlocality” of derivations in natural deduction: To control the correctness of inference steps in which assumptions can be discharged, we have to look higher up along derivation branches. (This will be crucial later with
the variable restrictions in quantifier rules.) In sequent calculus style, instead, each step of inference is local:

\[
\begin{align*}
\frac{A \supset (A \supset B) \vdash A \supset (A \supset B)}{A \supset (A \supset B), A \vdash A \supset B} & \quad \vdash E \\
A \supset (A \supset B), A \vdash B & \quad \vdash I \\
\vdash (A \supset (A \supset B)) \supset (A \supset B) & \quad \vdash I
\end{align*}
\]

In implication elimination, a rule with two premisses, the assumptions from the left of the turnstile are collected together. At the second implication elimination of the derivation, a second occurrence of \( A \) in the assumption part is produced. The trace of this repetition disappears, however, when assumptions are collected into sets.

The above system of introduction and elimination rules for \( \& \), \( \lor \), and \( \supset \), together with the rule of assumption by which an assumption can be introduced at any stage in a derivation, is the system of natural deduction for minimal propositional logic. If we add rule \( \bot \vdash E \) to it we have a system of natural deduction rules for intuitionistic propositional logic.

We obtain classical propositional logic by adding to the rules of intuitionistic logic a rule we call, in analogy to the law of excluded middle characteristic of classical logic in an axiomatic approach, the rule of excluded middle:

\[
\begin{array}{c}
[A] \\
\vdots \\
C
\end{array}
\quad \begin{array}{c}
[\sim A] \\
\vdots \\
C
\end{array}
\quad \frac{C}{C} \quad \text{Em}
\]

Both \( A \) and \( \sim A \) are discharged at the inference. The law of excluded middle, \( A \lor \sim A \), is derivable with the rule:

\[
\begin{align*}
\frac{[A]}{A \lor \sim A} \quad \frac{[\sim A]}{A \lor \sim A} & \quad \vdash l_1 \\
& \quad \vdash l_2 \\
\frac{A \lor \sim A}{A \lor \sim A} \quad \text{Em}
\end{align*}
\]

The rule of excluded middle is a generalization of the rule of indirect proof ("reductio ad absurdum"),

\[
\begin{array}{c}
[\sim A] \\
\vdots \\
A
\end{array}
\quad \frac{A}{Raa}
\]

The properties of the classical rules \( Em \) and \( Raa \) are presented in Chapter 8.
Rules of natural deduction can be categorized in a way similar to rules of proposition formation. This is based on the \textit{propositions-as-sets} principle, and leads to \textbf{type systems}. We think of a proposition \( A \) as being the same as its set of formal proofs. Each such proof can be called a \textit{proof-object} or \textit{proof term} to emphasize that this special notion of proof is intended. Instead of an assertion of the form \( \vdash A \) we have \( a : A \), \( a \) is a proof-object for \( A \). Rules of inference are categorized as functions operating on proof-objects.

Type-theoretical rules for proof-objects validate the BHK-explanations, by showing how proof-objects of compound propositions are constructed from proof-objects of their constituents. For example, the proof of an implication \( A \supset B \) is a function that converts an arbitrary proof of \( A \) into some proof of \( B \). In earlier times, the explanation of a proof of an implication \( A \supset B \) was described as “a method that converts proofs of \( A \) into proofs of \( B \);” and this was thought to be circular or at least ill-founded through its reference to an arbitrary proof of \( A \). However, in constructive type theory, the problem is solved.\footnote{The explanation was rejected on these grounds by Gödel (1941), for example. The solution was given, in philosophical terms, by Dummett (1975) and more formally by Martin-Löf (1975).} The meaning explanations first concern only “canonical proofs,” that is, the direct proofs of the forms given by the introduction rules. All other proofs, the “noncanonical” ones, are reduced to the canonical proofs through \textbf{computation rules} that correspond to the conversions in natural deduction. For this process to be well-founded, it is required that the conversion from noncanonical to canonical form terminate. These notions have deep connections to the structural properties of natural deduction derivations.

An exposition of type theory and its relation to natural deduction is given in Appendix B.

1.3. \textbf{From natural deduction to sequent calculus}

If our task is to derive \( A \supset B \), rule \( \supset \text{I} \) reduces the task to one of deriving \( B \) under the assumption \( A \). So we assume \( A \), but if \( B \) in turn is of the form \( C \& D \), rule \( \& \text{I} \) shows how the derivation of \( C \& D \) is reduced to that of \( C \) and \( D \). Thus we have to mentally decompose the goal \( A \supset B \) into subgoals, but there is no formal way to keep track of the process. It is as if we had to construct a derivation backwards.

\textbf{Sequent calculus} corrects the lack of guidance of natural deduction. It has a notation for keeping track of open assumptions; moreover, this is local: Each formula \( C \) has the open assumptions \( \Gamma \) that it depends on listed on the same line, as follows:

\[ \Gamma \Rightarrow C \]
Sequent calculus is a formal theory of the derivability relation. To make a
difference to writing $\Gamma \vdash C$, where the turnstile is a metalevel expression, not
part of the syntax as are the formulas, we use the now common symbol $\Rightarrow$. In
$\Gamma \Rightarrow C$, the left side $\Gamma$ is called the antecedent and $C$ the succedent.

As mentioned, the rules of natural deduction are schematic and show only
the active formulas, leaving implicit the set of remaining open assumptions. For
example, the rule of conjunction introduction can be written more completely as
follows, with a derivation of $A$ with open assumptions $\Gamma$ and a derivation of $B$
with open assumptions $\Delta$:

$$
\begin{array}{c}
\Gamma \quad \Delta \\
\vdots \\
A \quad B \\
\hline
A \& B
\end{array}
$$

Rule $\&I$ gives a derivation of $A \& B$ with open assumptions $\Gamma \cup \Delta$. With implication,
we have a derivation of $B$ from $A$ and $\Gamma$, and the introduction rule gives
a derivation of $A \supset B$ from $\Gamma$. The $E$-rules are similar; for example, disjunction
elimination gives a derivation of $C$ from $A \lor B$, $\Gamma$, $\Delta$, $\Theta$ if $C$ is derived from
$A$, $\Delta$ and from $B$, $\Theta$. The management of sets of assumptions was already made
explicit in the rules of natural deduction written in sequent calculus style. Sequent
calculus maintains the introduction rules thus written, but the treatment of
elimination rules is profoundly different.

The rules of sequent calculus are ordered in the same way as those of natu-
deduction, with the conclusion at the root. The introduction rules of natural
deduction become right rules of sequent calculus, where a comma replaces set-
theoretical union:

$$
\begin{array}{c}
\Gamma \Rightarrow A \\
\Delta \Rightarrow B \\
\hline
\Gamma, \Delta \Rightarrow A \& B
\end{array}
\quad
\begin{array}{c}
A, \Gamma \Rightarrow B \\
\hline
\Gamma \Rightarrow A \supset B
\end{array}
\quad
\begin{array}{c}
\Gamma \Rightarrow A \\
\hline
\Gamma \Rightarrow A \lor B
\end{array}
\quad
\begin{array}{c}
\Gamma \Rightarrow B \\
\hline
\Gamma \Rightarrow A \lor B
\end{array}
$$

Rule $\supset$ can also be read “root-first,” and in this direction it shows how the
derivation of an implication reduces to its components. Reduction here means
that the premiss is derivable whenever the conclusion is.

In Gentzen's original formulation of 1934–35, the assumptions $\Gamma$, $\Delta$, $\Theta$ were
finite sequences, or lists as we would now say. Gentzen had rules permitting the
exchange of order of formulas in a sequence. However, matters are simplified

---

2Use of the word "sequent" as a noun was begun by Kleene. His Introduction to Metamath-
ematics of 1952 (p. 441) explains the origin of the term as follows: "Gentzen says 'Sequenz',
which we translate as 'sequent', because we have already used 'sequence' for any succession of
objects, where the German is 'Folge'." This is the standard terminology now; Kleene's usage
has even been adopted to some other languages. But Mostowski (1965) for example uses the
literal translation "sequence."
if we treat assumptions as \textbf{finite multisets}, that is, lists with multiplicity but no order, and we shall do so from now on. Example 3 of Section 1.2 showed that if assumptions are treated simply as sets, control is lost over the number of times an assumption is made.

The elimination rules of natural deduction correspond to \textbf{left rules} of sequent calculus. In \&\&E, we have a derivation of $C$ from $A$, $B$ and some assumptions $\Gamma$, and we conclude that $C$ follows from $A\&B$ and the assumptions $\Gamma$. In sequent calculus, this is written as

$$\frac{A, B, \Gamma \Rightarrow C}{A\&B, \Gamma \Rightarrow C} \quad \text{(L&)}$$

The remaining two left rules are found similarly:

$$\frac{A, \Gamma \Rightarrow C \quad B, \Delta \Rightarrow C}{A \lor B, \Gamma, \Delta \Rightarrow C} \quad \frac{\Gamma \Rightarrow A \quad B, \Delta \Rightarrow C}{A \supset B, \Gamma, \Delta \Rightarrow C} \quad \text{(L\lor, L\supset)}$$

The formula with the connective in a rule is the \textbf{principal} formula of that rule, and its components in the premisses are the \textbf{active} formulas. The Greek letters denote possible additional assumptions that are not active in a rule; they are called the \textbf{contexts} of the rules.

In natural deduction elimination rules written in sequent calculus style, a formula disappears from the right; in sequent calculus, the same formula appears on the left. Inspection of sequent calculus rules shows what the effect of this change is.

\textbf{Subformula property:} All formulas in a sequent calculus derivation are subformulas of the endsequent of the derivation.

The usual way to find derivations in sequent calculus is a “root-first proof search.” However, in rules with two premisses, we do not know how the context in the conclusion should be divided between the antecedents of the premisses. Therefore we do not divide it at all but repeat it fully in both premisses. The procedure can be motivated as follows: If assumptions $\Gamma$ are permitted in the conclusion, it cannot do any harm to make the same assumptions elsewhere in the derivation. Rules R\&, L\lor, and L\supset can be modified into

$$\frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A\&B} \quad \frac{A, \Gamma \Rightarrow C \quad B, \Gamma \Rightarrow C}{A \lor B, \Gamma \Rightarrow C} \quad \frac{\Gamma \Rightarrow A \quad B, \Gamma \Rightarrow C}{A \supset B, \Gamma \Rightarrow C} \quad \text{(L\lor, L\supset)}$$

The preceding two-premiss rules had \textbf{independent contexts}; the above rules instead have \textbf{shared contexts}.\footnote{Lately some authors have called these “additive” and “multiplicative” contexts, but these are not as easy to remember.} It now follows that, given the endsequent to be
derived, once it is decided which formula of the endsequent is principal, the
premises are uniquely determined.

To show how derivations are found in sequent calculus, we derive the sequent

\[ \Rightarrow (A \supset (A \supset B)) \supset (A \supset B) \]

corresponding to Example 3 of Section 1.2:

\[
\begin{align*}
A \Rightarrow A & \quad B, A \Rightarrow B \\
A \Rightarrow A & \quad A \supset B, A \Rightarrow B \\
A \supset (A \supset B), A \Rightarrow B & \\
\Rightarrow (A \supset (A \supset B)) \supset (A \supset B)
\end{align*}
\]

Both instances of the two-premiss rule \( L \supset \) have the shared context \( A \). This root-
first proof search is not completely deterministic: The last step can be only \( R \supset \), but
above that, there are choices in the order of application of rules. Further, proof
search need not stop, but we stopped when we reached sequents with the same
formula in the antecedent and succedent. This situation corresponds to the rule
of assumption of natural deduction, by which we can start a derivation with any
formula \( A \) as assumption. The rule is given in sequent calculus in the form of a
logical axiom:

\[ A \Rightarrow A \]

In the above derivation, proof search ended in one case with a sequent of the form
\( A, \Gamma \Rightarrow A \), with a superfluous extra assumption. Its presence was caused by the
repetition of formulas in premises when shared contexts are used.

The \( \perp E \) rule of natural deduction gives a zero-premiss sequent calculus rule:

\[ \perp \Rightarrow C \]

Often this rule is also referred to as an axiom, but we want to emphasize its
class as a left rule and do not call it so.

Formally, a sequent calculus derivation is defined inductively: Instances of
axioms are derivations, and if instances of premisses of a rule are conclusions of
derivations, an application of the rule will give a derivation. Thus sequent calculus
derivations always begin with axioms or \( L \perp \). But we depart in two ways from
this “official” order of things:

First, note that the logical rules themselves are not derivations, for they have
sequents as premisses that need not be axioms. The combination of logical rules
likewise gives sequent calculus derivations with premisses. Each logical rule and
each combination is correct in the sense that, given derivations of the premisses,
the conclusion of the rule or of the combination becomes derivable.
Secondly, the usual root-first proof search procedure runs counter to the inductive generation of sequent calculus derivations. Proof search succeeds only when these two meet, i.e., when the root-first process reaches axioms or instances of $L\perp$.

We now come to the structural rules of sequent calculus. To derive the sequent $\Rightarrow A \supset (B \supset A)$ corresponding to Example 2 in Section 1.2, we use a rule of weakening that introduces an extra assumption in the antecedent:

$$\frac{\Gamma \Rightarrow C}{A, \Gamma \Rightarrow C}^{wk}$$

The rule is sometimes called “thinning.” The derivation of Example 2 is

$$\frac{A \Rightarrow A}{A, B \Rightarrow A}^{wk}$$
$$\frac{A \Rightarrow B \supset A}{\Rightarrow A \supset (B \supset A)}^{R\supset}$$

The derivation illustrates the role of weakening: Whenever there is a vacuous discharge in a natural deduction derivation, there is in a corresponding sequent calculus derivation an instance of a logical rule with an active formula that has been introduced in the derivation by weakening.

As noted, our example of proof search in sequent calculus led to a premiss that was not an axiom of the form $A \Rightarrow A$, but of the form $A, \Gamma \Rightarrow A$. These more general axioms are obtained from $A \Rightarrow A$ by repeated application of weakening. If instead we permit instances of axioms as well as the $L\perp$ rule to have an arbitrary context $\Gamma$ in the antecedent, there is no need for a rule of weakening in sequent calculus.

Above we gave a derivation of the sequent corresponding to Example 3 of Section 1.2 using rules with shared contexts. We give another derivation, this time with the earlier rules that have independent contexts. A rule of contraction is now needed:

$$\frac{A, A, \Gamma \Rightarrow C}{A, \Gamma \Rightarrow C}^{ctr}$$

With this rule and axioms of the form $A \Rightarrow A$, the derivation is

$$\frac{A \Rightarrow A}{A \Rightarrow A}^{L\supset}$$
$$\frac{A \supset (A \supset B), A, \Gamma \Rightarrow B}{\Rightarrow (A \supset (A \supset B)) \supset (A \supset B)}{R\supset}$$

Contrary to the derivation with shared contexts, a duplication of $A$ is produced
on the fourth line from below. The meaning of contraction can be explained in terms of natural deduction: Whenever there is a multiple discharge in natural deduction, there is a contraction in a corresponding sequent calculus derivation.

If assumptions are treated as sets instead of multisets, contraction is in a way built into the system and cannot be expressed as a distinct rule.

As with weakening, the rule of contraction can be dispensed with, by use of rules with shared contexts and some additional modifications.

In Chapter 8 we show in a general way that weakening and contraction amount to vacuous and multiple discharge, respectively, in natural deduction, whenever the weakening or contraction formula is active in a derivation. Without this condition, weakening and contraction are purely formal matters produced by the separation of discharge of assumptions into independent structural and logical steps in sequent calculus.

We now come to the last and most important general rule of sequent calculus. In natural deduction, if two derivations \( \Gamma \vdash A \) and \( A, \Delta \vdash C \) are given, we can join them together into a derivation \( \Gamma, \Delta \vdash C \), through a substitution. The sequent calculus rule corresponding to this is cut:

\[
\begin{array}{c}
\Gamma \Rightarrow A \\
A, \Delta \Rightarrow C
\end{array} \quad \frac{}{\Gamma, \Delta \Rightarrow C} ^ \text{Cut}
\]

Often cut is explained as follows: We break down the derivation of \( C \) from some assumptions into “lemmas,” intermediate steps that are easier to prove and that are chained together in the way shown by the cut rule. In Chapter 8 we find a somewhat different explanation of cut: It arises, in terms of natural deduction, from non-normal instances of elimination rules. This points to an important analogy between normal derivations in natural deduction and cut-free derivations in sequent calculus, an analogy that will be made precise in Chapter 8.

As explained in Section 1.2, there is in natural deduction a series of concepts from the existence of normal form to strong normalization and uniqueness of normal form. In systems of sequent calculus, it is possible that two derivations \( \Gamma \Rightarrow A \) and \( A, \Delta \Rightarrow C \) are cut-free, but the derivation \( \Gamma, \Delta \Rightarrow C \) obtained by cut need not, in general, have any such form. (This would correspond to the inexistence of normal form in natural deduction.) In the contrary case, we say that a system of sequent calculus is closed with respect to cut: If there is a derivation of the sequent \( \Gamma \Rightarrow C \) that uses the rule of cut, there exists also a derivation without cut. A typical way of proving closure under cut is to show the completeness of a system that does not have the cut rule: All correct sequents are derivable in the system so that the addition of cut does not add any new derivable sequents. Next there is the notion of elimination of cut in which a procedure for the actual elimination of cuts in a derivation is given. It corresponds to normalization in natural deduction, but it is typical of sequent calculi that they do not admit of
properties corresponding to strong normalization or uniqueness of normal form: in terms of sequent calculi, termination of cut elimination in any order whatsoever and uniqueness of the cut-free form.

An alternative formulation of the rule of cut is obtained if rule $R\supset$ is applied to its right premiss $A, \Delta \Rightarrow C$. The derivation

$$
\begin{array}{c}
A, \Delta \Rightarrow C \\
\Gamma \Rightarrow A \\
\Delta \Rightarrow A \supset C
\end{array}
\frac{}{
\Gamma, \Delta \Rightarrow C}
$$

shows how cut is replaced by rule $R\supset$ and a sequent calculus version of modus ponens.

Weakening, contraction, and cut are the usual structural rules of sequent calculus. Cut has the effect of making a formula disappear during a derivation so that it need not be a subformula of the conclusion, whereas none of the other rules do this. If we wanted to determine whether a sequent $\Gamma \Rightarrow C$ is derivable, by using cut we could always try to reduce the task into $\Gamma \Rightarrow A$ and $A, \Gamma \Rightarrow C$ with a new formula $A$, with no end.

A main task of structural proof theory is to find systems that do not need the cut rule or use it in only some limited way. But note that contraction can be as “bad” as cut as concerns a root-first search for a derivation of a given sequent: Formulas in antecedents can be multiplied with no end if contraction cannot be dispensed with.

Two main types of sequent calculi arise: those with independent contexts, similar in many respects to calculi of natural deduction, and those with shared contexts, useful for proof search. Gentzen’s original (1934–35) calculi for intuitionistic and classical logic had shared contexts for $R\&$ and $L\lor$ and independent ones for $L\supset$. Further, the left rule for $\&$ (as well as the $R\lor$ rule in the classical case) was given in the form of two rules

$$
\begin{array}{c}
A, \Gamma \Rightarrow C \\
A \& B, \Gamma \Rightarrow C
\end{array}
\frac{}{
A \& B, \Gamma \Rightarrow C}
$$

that do not support proof search: It need not be the case that $A, \Gamma \Rightarrow C$ is derivable even if $A \& B, \Gamma \Rightarrow C$ is. The single $L\&$ rule we use is due to Ketonen (1944). He also improved the classical $R\lor$ rule in an analogous way and found a classical $L\supset$ rule with shared contexts. With these changes, the sequent calculus for classical propositional logic is invertible: From the derivability of a sequent of any of the forms given in the conclusions of the logical rules, the derivability of its premisses follows. Starting with the endsequent, decomposition by invertible rules gives a terminating method of proof search for classical propositional logic.

For intuitionistic logic, a sequent calculus with shared contexts was found by Kleene (1952). The rule of cut can be eliminated in calculi with independent as
well as shared contexts. In calculi of the latter kind, also weakening and contraction can be eliminated, so that derivations contain logical rules only. Chapters 2–4 are mainly devoted to the development and study of such calculi. Calculi with independent contexts are studied in Chapter 5.

1.4. THE STRUCTURE OF PROOFS

Given a system of rules G, we say that a rule with premisses S₁, . . . , Sₙ and conclusion S is admissible in G if, whenever an instance of S₁, . . . , Sₙ is derivable in G, the corresponding instance of S is derivable in G. Structural proof theory has as its first task the study of admissibility of rules such as weakening, contraction, and cut. Our methods for establishing such results will be thoroughly elementary: In part we show that the addition of a structural rule has no effect on derivability (as for weakening), or we give explicit transformations of derivations that use structural rules into ones that do not use them (as for cut). A major difficulty is to find the correct rules in the first place. Even if the proof methods are all elementary, the proofs often depend on the right combination of many details and are much easier to read than write.

If the cut rule has been shown admissible for a system of rules, we see by inspection of all the rules of inference that no formula disappears in a derivation. Thus cut-free derivations have the subformula property: Each formula in the derivation of a sequent Γ ⇒ C is a subformula of this endsequent. Later we shall relax on this a bit, by letting atomic formulas disappear, and then the subformula property becomes the statement that each formula in a derivation is a subformula of the endsequent or an atomic formula. Such a weak subformula property is still adequate for structural proof-analysis.

Standard applications of cut elimination include elementary syntactic proofs of consistency, the disjunction property for intuitionistic systems, interpolation theorems, and so on. For the first, assume a system to be inconsistent, i.e., assume the sequent ⇒ ⊥ is derivable in it. Each logical rule adds a logical constant, and the axioms and weakening and contraction are rules that have formulas in the antecedent. Therefore there cannot be any derivation of ⇒ ⊥; a cut-free system is consistent. Similarly, assuming that ⇒ A ∨ B is derivable in a system of rules, it can be the case that the only way by which it can be concluded is by the rules for right disjunction. Thus either ⇒ A or ⇒ B can be derived, and we say that the system of rules has the disjunction property. If a system is both cut-free and contraction-free, it can have the property that the premisses are proper parts of the conclusion, i.e., at least some formula is reduced to a subformula. In this case, we have a root-first proof search resulting in a tree that terminates. If the leaves of the tree thus reached are axioms or instances of L⊥, we can read it top-down as a derivation of the endsequent. But to show that a sequent is underivable, we have to be able to survey all possible derivations. For example, assume that
$\Rightarrow P \lor \sim P$ is derivable in a cut-free intuitionistic system. Then the last rule is one of the two right disjunction rules, and either $\Rightarrow P$ or $\Rightarrow \sim P$ is derivable. No logical rule concludes $\Rightarrow P$, and if $\Rightarrow \sim P$ were derivable, the last rule would have had to be $R\supset$. Again, no logical rule concludes the premiss $P \Rightarrow \bot$.

Above we found a way that led to the rules of sequent calculus from those of natural deduction. Often the structure of cut-free sequent calculus derivations is seen more clearly if it is translated back into natural deduction. This can be made algorithmic, as shown in Chapter 8. Not all sequent calculus derivations can be translated, but only those that do not have “useless” weakening or contraction steps. The translation is such that the order of application of logical rules is reflected in the natural deduction derivation. The meaning of a cut-free derivation is that all major premisses of elimination rules turn into assumptions.

The connection between sequent calculus and natural deduction is straightforward for single succedent sequent calculi, i.e., those with just one formula in the succedent to the right of the sequent arrow. However, there are also systems with a whole multiset as succedent. It can be shown that systems of intuitionistic logic are obtained from classical multisuccedent systems by some innocent-looking restrictions on the succedents. In Chapter 5 we show that the converse is also true, at least for propositional logic: We obtain classical logic from intuitionistic single succedent sequent calculus by the addition of a suitable rule corresponding to the classical law of excluded middle.

Most of the research on sequent calculus has been on systems of pure logic. Considering that the original aim of proof theory was to show the consistency of mathematics, this is rather unfortunate. It is commonly believed that there is nothing to be done: that the main tool of structural proof theory, cut elimination, does not apply if mathematical axioms are added to the purely logical systems of derivation of sequent calculus. In Chapter 6 we show that these limitations can be overcome. A simple example of the failure of cut elimination in the presence of axioms is given by Girard (1987, p. 125): Let the axioms have the forms $\Rightarrow A \supset B$ and $\Rightarrow A$. The sequent $\Rightarrow B$ is derived from these axioms by

$$
\frac{
\Rightarrow A \supset B \quad A \supset B, A \Rightarrow B \quad}{\Rightarrow A}
A \Rightarrow A \quad B \Rightarrow B \quad L\supset
\frac{
A \Rightarrow B \quad}{\Rightarrow B}
A \Rightarrow B \quad Cut
$$

Inspection of sequent calculus rules shows that there is no cut-free derivation of $\Rightarrow B$, which leads Girard to conclude that “the Hauptsatz fails for systems with proper axioms” (ibid.). More generally stated, the cut elimination theorem does not apply to sequent calculus derivations with premisses.

We shall give a way of adding axioms to sequent calculus in the form of nonlogical rules of inference and show that cut elimination need not be lost by such addition. This depends critically on formulating the rules of inference in a
particular way. It then follows that the resulting systems of sequent calculus are both contraction- and cut-free. A limitation, not of the method, but one that is due to the nature of the matter, is that in constructive systems there will be some special forms of axioms, notably \((P \supset Q) \supset R\), that cannot be treated through cut-free rules. For classical systems, our method works uniformly. Gentzen's original subformula property is lost, but typical consequences of that property, such as consistency or the disjunction property for constructive systems, usually follow from the weaker subformula property.

To give an idea of the method, consider again the above example. With \(P\) and \(Q\) atomic formulas and \(C\) an arbitrary formula, \(P \supset Q\) is turned into a rule by requiring that if \(C\) follows from \(Q\), then it follows from \(P\) and \(P\) is turned into a rule by requiring that if \(C\) follows from \(P\), then \(C\) follows:

\[
\frac{Q \Rightarrow C}{P \Rightarrow C} \quad \frac{P \Rightarrow C}{\Rightarrow C}
\]

The sequent \(\Rightarrow Q\) now has the cut-free derivation

\[
\frac{Q \Rightarrow Q}{P \Rightarrow Q} \Rightarrow Q
\]

The method of converting axioms into cut-free systems of rules has many applications in mathematics; for example, it can be used in syntactic proofs of consistency and mutual independence for axiom systems. If we use classical logic, we can convert a theorem to be proved into a finite number of sequents that have no logical structure but only atomic formulas and falsity. By cut elimination, their derivation uses only the nonlogical rules, and a very strong control on structure of derivations is achieved. In typical cases such as affine geometry, an axiom can be proved underviable from the rest of an axiom system by showing its underviability by the rules corresponding to the latter.

The aim of proof theory, as envisaged by Hilbert in 1904, was to give a consistency proof of arithmetic and analysis and thereby to resolve the foundational problems of mathematics for good. There had been earlier consistency proofs, such as those for non-Euclidean geometries, in which a model was given for an axiom system. However, such proofs are relative: They assume the consistency of the theory in which the model is given. Hilbert's aim instead was an absolute consistency proof, carried through by elementary means. The results of Gödel in 1931 are usually taken to show such proofs to be an impossibility as soon as a system contains the principles of arithmetic. However, we shall see in Chapter 6 that, when this is not the case, purely syntactic and elementary consistency proofs can be obtained as corollaries to cut elimination.

A whole branch of logical research is devoted to the study of intermediate logical systems. These are by definition systems that stand between intuitionistic
and classical logic in deductive power. In Chapter 7, we shall study the structure of proofs in intermediate logical systems by presenting them as extensions of the basic intuitionistic calculus. One method of extension follows the model of extending this calculus by the rule of excluded middle. Such extension works perfectly for the logical system obeying the weak law of excluded middle, \( \sim A \lor \sim \sim A \). A limit is reached here, too, for in order to have a subformula property, the characteristic law of an intermediate logic is restricted to instances of the law with atomic formulas, as for the law of excluded middle. If the law for arbitrary formulas cannot be proved from the law for atoms, there is no good structural proof theory under this approach. Such is the case for Dummett logic, characterized by the law \( (A \supset B) \lor (B \supset A) \). Another method that has been used is to start with the multisuccedent intuitionistic calculus and to relax the intuitionistic restriction on the \( R \supset \) rule. This approach will lead to a satisfactory system for Dummett logic.

Our approach to structural proof theory is mainly based on contraction- and cut-free sequent calculi. However, we also present, in Chapter 5, calculi in which weakening and contraction are explicit rules and only cut is eliminated. The sequent calculus rules of the previous section are precisely the propositional and structural rules of the first such calculus, in Section 5.1(a). Further, we also present a calculus in which there is no explicit weakening or contraction, but these are built into the logical rules. This calculus, studied in Section 5.2, can be described as a "sequent calculus in natural deduction style." Sequent calculi with independent contexts are useful for relating derivations in sequent calculus to derivations in natural deduction. The use of special elimination rules in natural deduction brings problems that vanish only if the general elimination rules are taken into use. In Chapter 8 we show that this change will give an isomorphism between sequent calculus derivations and natural deduction derivations. The analysis of proofs by means of natural deduction can often provide insights it would be hard to obtain by the use of sequent calculus only.

Notes to Chapter 1

The definition of languages through categorial grammars, and predicate logic especially, is treated at length in Ranta’s *Type-Theoretical Grammar*, 1994. A discussion of logical systems from the point of view of constructive type theory is given in Martin-Löf’s *Intuitionistic Type Theory*, 1984, but see also Ranta’s book for later developments.

An illuminating discussion of the nature of logical rules and the justification of introduction rules in terms of constructive meaning explanations is given in Martin-Löf (1985). Dummett’s views on these matters are collected in his *Truth & Other Enigmas* of 1978.

Our treatment of the elimination rules of natural deduction for propositional logic comes from von Plato (1998) and differs from the usual one that recognizes only the special elimination rules, as in Gentzen’s original paper *Untersuchungen über*
Das logische Schliessen (in two parts, 1934–35) or Prawitz’ influential book *Natural Deduction: A Proof-Theoretical Study* of 1965. The change is due to our formulation of the inversion principle in terms of arbitrary consequences of the direct grounds of the corresponding introduction rule, instead of just these direct grounds. The general elimination rule for conjunction is presented in Schroeder-Heister (1984). Chapter 8 will show what the effect of general elimination rules is for the structure of derivations in natural deduction.

Natural deduction in sequent calculus style is used systematically in Dummett’s book *Elements of Intuitionism* of 1977.

Our way of obtaining classical propositional logic from the intuitionistic one uses the rule of excluded middle. It appears in this form, as a rule for arbitrary propositions, in Tennant (1978) and Ungar (1992), but the first one to propose the rule was Gentzen (1936). The rule has not been popular, for the obvious reason that it does not have the subformula property. Prawitz (1965) uses the rule of indirect proof and shows that its restriction to atomic formulas will give a satisfactory normal form and subformula property for derivations in the \( \lor \)-free fragment of classical propositional logic. We restrict in Chapter 8 the rule of excluded middle to atomic formulas and show that this gives a complete system of natural deduction rules and a full normal form for classical propositional logic. We also show that the rule can be restricted to atoms of the conclusion, thereby obtaining the full subformula property.

The long survey article by Prawitz, *Ideas and results in proof theory*, 1971, offers valuable insights into the development of structural proof theory. The notes to the chapters of Troelstra and Schwichtenberg’s *Basic Proof Theory*, 1996, also contain many historical comments. Feferman’s *Highlights in proof theory*, 2000, discusses Hilbert’s program, sequent calculi, and the proof theory of arithmetic. Finally, the life story of the founder of structural proof theory is given in Menzler-Trott’s *Gentzen’s Problem: Mathematische Logik im nationalsozialistischen Deutschland*, 2001.