Peter Schroeder-Heister

Structural Frameworks with Higher-Level Rules

Philosophical Investigations on the Foundations of Formal Reasoning

Universität Konstanz
Fachgruppe Philosophie
Postfach 5560
7750 Konstanz, F.R.G.

July 1987

Preface

I should like to thank Kosta Došen, Lars Hallnäs, Per Martin-Löf, Dag Prawitz, Stephen Read and Neil Tennant for helpful comments and discussions on topics related to this work.

Drafts of Chapters 5 and 6 were written during a research stay at the Department of Logic and Metaphysics of the University of St. Andrews in Spring 1985. I am grateful to the Fritz Thyssen Stiftung for supporting this stay with a travel grant.

I should like to thank my colleagues, especially Gereon Wolters, for contributing to a congenial and stimulating atmosphere at the Department of Philosophy here in Konstanz. In particular, I wish to express my appreciation to Jürgen Mittelstraß for continuously supporting me in my work and giving me the opportunity to pursue the research interests that resulted in this study.

Finally, a note of thanks to Ron Feemster and especially to Raoul Eshelman for revising the English manuscript.

Added 2003:
I am very grateful to Thomas Piecha for converting the original file into LaTeX.
He carried out this tedious task with extreme diligence.

Concerning the content of this work I should mention that the logical system of positive relevance logic arrived at in Chapter 5 is not **R**, as claimed on p. 48, but the system **RM** of relevance logic with the mingle rule.

Contents

Introduction

The aim of this study is to extend Gentzen's notion of structure by proposing a general theory of reasoning with rules. Its background is a certain tradition of proof theory whose basic tenet is that investigating formal reasoning and therefore proofs is essentially a philosophical rather than mathematical venture, even when technical methods are used.

Underlying this approach is the idea that logic as a philosophical discipline with foundational claims is primarily a theory of *logical inference*. According to this view, the basic task of logic is to study how propositions are *established* (possibly from other propositions as assumptions) on purely formal grounds. Logical proofs receive an *epistemological* interpretation: They are thought to represent a basic activity by means of which humans acquire knowledge. Proofs are correct not by virtue of an external notion of logical consequence, but because each single inference step is directly evident by virtue of the meanings of the logical constants involved in it. Thus, whether or not a proposition is a logical consequence of other propositions is a *result* of inference steps.

The epistemological understanding of "proof" was standard in the philosophical tradition from Aristotle and Euclid to Frege. In 20th century logic, in particular within Hilbert-style proof theory, it shifted to the idea of formal proofs or derivations. A formal proof can be considered an arrangement of symbols generated by a fixed set of primitive rules. These rules indicate which strings of symbols one may begin with and which steps one may use to move from the symbols already generated to other ones. Consequently, Hilbert considered proof theory a new branch of mathematics, called "metamathematics" (see Hilbert 1923).

Within proof theory, this formalist position was implicitly challenged by Gentzen's work, in particular by his "Investigations into Logical Deduction" (Gentzen 1935a). Although Gentzen understood himself as working within Hilbert's program of justifying classical mathematics by means of consistency proofs (see Gentzen 1938), his way of developing systems for formalizing mathematics has implications which transcend this program. In particular, Gentzen did not merely consider his calculus of natural deduction as a useful tool of metamathematics, but also relied on considerations concerning the meanings of the logical constants (Gentzen 1935a, p. 187-188). He claimed that natural deduction—with its characteristic systematics of introduction and elimination rules—reflected "real" reasoning (ibid.). In doing so, Gentzen met the central requirement of an epistemological reading of proofs: that every inference step gives us a certain amount of "insight". This tradition was continued by Prawitz (1965), who formulated the systematics of Gentzen's rules of natural deduction in the form of an "inversion principle" related to Lorenzen's (1955) similar principle of the same name. The traditional philosophical sense of "proof" was then explicitly re-established in proof theory by Prawitz (1973, 1974), who also pointed out the relationship between proof theory and the theory of meaning in Dummett's sense (see Dummett 1975, Prawitz 1977, 1981, 1987, and Tennant 1987). About the same time Martin-Löf started developing his intuitionistic theory of types (see Martin-Löf 1975) as a foundational theory of constructive mathematics based on epistemological and semantical considerations.

This philosophical conception of proof theory does not imply that we may no longer work in the context of formal systems if we want to claim philosophical significance for our results. It means only that we must make the inference rules and inference schemata of our formal systems intuitively plausible in such a way that derivations in a formal system can be considered codifications of arguments. In many cases it is even preferable to first consider formal systems rather than to immediately pass on to "real" reasoning, since formal systems require certain standards of precision which otherwise cannot always be reached.[1]

When formal proofs are presented in a sequent-style framework, not all inferences will refer to the logical form of expressions, i.e. to the way expressions are composed from subexpressions by means of logical constants. Inferences of this kind were called "structural" by Gentzen (1935a). Examples are thinning or contraction, which can be written as follows by using X for finite lists of formulae, A and B for formulae and the turnstile "$\vdash$" as the sequent sign:

$$\frac{X \vdash A}{B, X \vdash A} \qquad\qquad \frac{B, B, X \vdash A}{B, X \vdash A}\,.$$[2]

To motivate our extension of Gentzen's notion of structure we consider the following three inference steps, which may be part of a sequent calculus of intuitionistic propositional logic (as in Gentzen 1935b):

$$(*1)\quad \frac{X, A \vdash B}{X \vdash A \supset B}\,(\supset\text{-I})$$

$$(*2)\quad \frac{X \vdash A \quad X \vdash B}{X \vdash A \,\&\, B}\,(\&\text{-I})$$

$$(*3)\quad \frac{X \vdash A \vee B \quad X, A \vdash C \quad X, B \vdash C}{X \vdash C}\,(\vee\text{-E})$$

Normally these steps would be regarded as instances of *three specific inference schemata*: the first as an instance of $\supset$-introduction, the second of &-introduction, and the third of

[1]It should be mentioned in this context that central epistemological and semantical principles of Prawitz's and Martin-Löf's *philosophical* theories result from re-interpreting *technical* notions of the theory of natural deduction systems.

[2]Only sequents with one formula in the succedent are taken into consideration here.

$\vee$-elimination. Since these inference schemata refer to particular logical constants, they are usually called *logical* inference schemata, as opposed to structural ones.

Our proposal is to consider ($*1$), ($*2$) and ($*3$) as instances of *one general inference schema* which governs the application of rules. Such a schema can be written as

$$(**) \qquad (Y_1 \Rightarrow A_1), \ldots, (Y_n \Rightarrow A_n) \Rightarrow D \, \frac{X, Y_1 \vdash A_1 \;\; \ldots \;\; X, Y_n \vdash A_n}{X \vdash D} \;,$$

where $(Y_1 \Rightarrow A_1), \ldots, (Y_n \Rightarrow A_n) \Rightarrow D$ is the rule to be applied. In the case of ($*1$) it has the form $(A \Rightarrow B) \Rightarrow A \supset B$, in the case of ($*2$) the form $A, B \Rightarrow A \,\&\, B$ and in case ($*3$) the form $(A \vee B, (A \Rightarrow C), (B \Rightarrow C)) \Rightarrow C$. In general, the intended meaning of the rule $(Y_1 \Rightarrow A_1), \ldots, (Y_n \Rightarrow A_n) \Rightarrow D$, which is made explicit by ($**$), is the following: If for each i $(1 \leq i \leq n)$ we have established A_i under the assumptions Y_i and X, then we may pass over to D (under the same assumptions X).

In our proposal we strictly distinguish between *rules* on the one hand, which are a certain sort of syntactic objects written linearly by using the rule arrow "$\Rightarrow$", and *inference schemata* on the other, which are written two-dimensionally by using an inference line and from which derivation trees can be constructed. This means that we refrain from identifying rules with *specific* inference schemata and instead consider the application of rules as something proceeding according to certain *general* inference schemata such as ($**$). Since these general inference schemata do not refer to the internal (logical) form of formulae, they may be called *structural* and put in parallel to thinning or contraction. Thus in our systems, *all* inference schemata are structural inference schemata. Since the specific content of a given system is represented by certain primitive rules, we may change a system by altering its primitive *rules*, whereby the *inference schemata* remain fixed. For example, suppose the disjunction-free fragment of intuitionistic logic is given. Then a full system of intuitionistic logic could be obtained without altering any inference schema by simply adding the primitive rules $A \Rightarrow A \vee B$, $B \Rightarrow A \vee B$ and $(A \vee B, (A \Rightarrow C), (B \Rightarrow C)) \Rightarrow C$. This means in particular that if we can prove something that relies only on the inference schemata given, it holds for all systems which vary only in the choice of primitive rules.

Of course, it is not possible to formulate a general concept of "rule" and corresponding general inference schemata for rule application which cover everything logic considers to be an inference. Nonetheless, certain notions of rules with corresponding structural inference schemata can be developed that allow us to treat certain branches of logic in a uniform manner. In particular, we shall discuss systems of first-order logic with an arbitrary number of logical operators (Chs. 1 to 3), atomic inference systems used in logic programming (Ch. 4), systems of relevance logic (Ch. 5) and logical systems of the kind proposed by Martin-Löf (Ch. 6). Thus we speak of "structural frameworks" permitting the treatment of certain types of logical and non-logical systems. More precisely, we understand by a *structural framework* a concept of "rule" along with a set of inference schemata that define how rules are handled within derivations. We shall show how different ideas about formal reasoning may lead to different underlying structural conceptions.

Once we have introduced rules as special objects which are treated in derivations according to certain structural inference schemata, there is no reason why we should consider only *primitive* rules of inference. In addition to primitive rules, which are so to speak given from "without", we may allow for rules as assumptions which occcur in sequents to the left of the turnstile. Since even formulae can be considered limiting cases of rules (viz. of rules which permit one to establish the formula itself), the result is a system in which all elements of a sequent are rules. This leads to a generalized concept of a rule: If a rule can occur in the antecedent of a sequent, it should also be allowed to occur as the premise of another rule. The latter rule is then of greater complexity than rules in the usual sense. By iterating rule formation in this way, we obtain a general concept of higher-level rules. Correspondingly, structural schemata can be developed which govern reasoning with such rules in a plausible way.

Rules of higher levels extend the available means of expression in a significant way. This has particular relevance for the definition of primitive rules for logical constants. If a logical constant is characterized by introduction and elimination rules, and the premises of its introduction rules are considered systems of rules of potentially higher levels, then we obtain a general pattern for logical rules which yields a uniform perspective on a variety of logical systems.[3] The range of logical constants which is covered in this way depends on the distinctions which are available at the structural level, i.e. on the possibilities for formulating different premises of introduction rules. For example, if a kind of negation is available at the structural level, then a corresponding logical constant of negation can be defined; and if a rule arrow which structurally expresses a certain "relevant" connection is at one's disposal, then a logical constant for relevant implication can be given introduction and elimination rules. This comes close to Došen's thesis, claimed with respect to a structural apparatus somewhat different from ours, that logical constants serve as "punctuation marks" expressing "structural features of deduction" in the object language (Došen 1988, see also 1986b).[4]

Frameworks with rules of higher levels are not only important for the definition of logical constants. They are also a useful extension of calculi for atomic formulae. In particular, they lead to a natural extension of definite Horn clause programs and therefore of the programming language PROLOG, the theory of which can be developed within a theory of atomic systems. Although such an extension can also be described in purely logical terms (by replacing the rule arrow "$\Rightarrow$" by implication "$\supset$" and the comma by conjunction "&"), the proof-theoretic approach based on higher-level rules is both technically more elegant and intuitively more plausible.

Although the term "structural" was introduced by Gentzen in the context of sequent-style calculi, it also makes sense for systems of natural deduction. For example, if we regard a derivation of A that depends on a set of assumptions X as a derivation of A

[3]This is of particular interest for the implementation of logical systems on a computer, since it relieves one of the necessity of considering each particular system as a unity of its own for which specific programming efforts would have to be expended. The most recent approach to a universal framework for the definition of logical (including type-theoretical) systems is the Edinburgh LF ("logical framework"), into which some of the ideas presented here concerning the general form of elimination rules have been incorporated (see Harper, Honsell and Plotkin 1987).

[4]In this investigation, however, we do not want to make general claims about delimiting logical constants from non-logical ones, since this is a subject of its own.

from any superset of X (see Prawitz 1965), we are implicitly using a structural principle which corresponds to thinning in sequent-style systems. This is even more obvious when we extend the notion of structure to include inference schemata for rule application. In natural deduction style, the inference schema $(**)$ would read as

$$(Y_1 \Rightarrow A_1), \ldots, (Y_n \Rightarrow A_n) \Rightarrow D \qquad \frac{\begin{array}{ccc} [Y_1] & & [Y_n] \\ \vdots & & \vdots \\ A_1 & \ldots & A_n \end{array}}{D} \ ,$$

where the square brackets surrounding the Y_i indicate that these assumptions may be discharged.[5] Nevertheless, we shall work only with sequent-style systems in the following, since in sequents, which always display the assumptions under which an assertion is made, structural features are more explicit.[6] This is particularly important for the application of higher-level rules in logic programming, in the theory of relevance logic and in Martin-Löf's theory, which are central subjects of this study. For all of these topics it is essential that the structural assumptions are precisely stated.

Of course, introducing higher-level rules and the corresponding inference schemata which govern their use is not the only way of extending Gentzen's idea of structural inferences. Another approach is to iterate sequent formation, thus yielding sequents of higher levels, whereby antecedents and succedents themselves contain sequents. This proposal, which has been set forth by Došen (1980), leads both to a natural proof-theoretic treatment of modal logics (Došen 1985, 1986a) and to a very plausible philosophical approach to the problem of the logicality of "logical" constants (see Došen 1988).[7]

Another direction in which our structural frameworks could be extended is a deeper treatment of negation. In the context of the present investigation, all negations considered are defined or are definable by constants expressing absurdity. Using von Kutschera's (1969) terminology, this means that negation is always understood in an "indirect" way as "leading to something that is absurd". An interesting alternative is von Kutschera's "direct" negation, which could be obtained by additionally introducing the concept of refutation rules relating to the denials of sentences. A theory of this type of negation was developed by von Kutschera (1969, 1985) in a framework which, as far as its positive propositional fragment is concerned (i.e. the fragment without the denial operator and without quantifiers, see von Kutschera 1968), is closely related to ours. Von Kutschera uses the iteration of an arrow "$\rightarrow$" which is considered not as a logical constant but as a structural sign in our sense. This arrow functions both as the sequent sign (the turnstile "$\vdash$" in our notation) and as the rule arrow (our "$\Rightarrow$"). Although von Kutschera uses a slightly different justification for his system, referring to the idea of an infinite hierarchy

[5]For a theory of higher-level rules in natural deduction systems see Schroeder-Heister (1981, 1984a, 1984b).

[6]A sequent-style formulation of Schroeder-Heister (1984a) has already been proposed by Avron (1986).

[7]The combination of both approaches—namely sequents of higher levels together with rules of higher levels—is the subject of a joint project now being undertaken by Kosta Došen and myself.

of calculi, the formalism at which he finally arrives is similar to our system SC3+ defined in Ch. 1 § 2.

A further limitation in our approach that might be removed in subsequent work is its restriction to sequent calculi whose sequents have only one element in the succedent. With respect to our concept of higher-level rules, this restriction means that we consider only rules with single conclusions. A framework with multiple-conclusion rules that is based on the idea of a structural analogue to logical disjunction might lead to a more adequate treatment of systems of classical logic.[8] In our structural framework, classical logic is dealt with by proposing a structural analogue to classical "reductio ad absurdum". With multiple-conclusion sequents and rules, classical logic could be obtained in a more direct way. However, it would then not be a trivial task to provide structural disjunction with an intuitively plausible meaning.

Finally, in an even more abstract setting than our structural frameworks, one could discuss whether there is some generic notion of "rule" which encompasses various structural notions of rules, especially the two types of rules we are going to define (viz. extensional and intensional ones, the latter in connection with relevance logic). Such an investigation, which would unify several structural frameworks in a "superstructure", would correspond to Belnap's "display logic" (Belnap 1982), which develops generic notions of binary structural combinations (corresponding to the comma), structural negation, and nullary structural constants (representing truth or falsity). Different structural requirements for these generic combinations lead to different families of structural combinations, which in turn lead to different families of logical constants (such as Boolean, modal or relevant constants).

As should be clear from these introductory remarks, our principal aim is a theoretical one. That is, we do not claim that our sequent calculi, which provide a general picture of how rules are handled in derivations, are best suited for carrying out derivations for a *specific* system. On the contrary, when working with a limited number of primitive rules, it may be more convenient to identify rules (as is usual) with the inferences or inference schemata according to which they are applied. For us it is essential that reasoning with rules has structural features which *can* be—but do not always *have* to be—made explicit.[9]

The guiding idea governing our investigations is to give formal reasoning a uniform representation by enriching its structural aspects. In Chapter 1 we develop several sequent-style formalisms for reasoning with higher-level rules and prove their equivalence. We also introduce rules with generality, which can express eigenvariable conditions. In Chapter 2 this framework is used to characterize logical constants by means of introduction and elimination rules which follow a general pattern. In order to include quantifiers in our approach, logical constants are considered operators with predicate terms as arguments. In other words, variable-binding is connected with λ-abstraction and not with

[8]The term "multiple-conclusion logic" is due to Shoesmith and Smiley (1978).

[9]This also holds for other extensions of Gentzen's structural apparatus mentioned above. For example, for actually carrying out derivations in the modal logic S4, one would not normally use Belnap's display logic or Došen's second-level sequents, although these concepts give us much insight into the structure underlying S4 and are very useful for meta-theoretical investigations of this system.

quantification. The standard constants of intuitionistic logic play a special role since they suffice to explicitly define all other logical constants whose characteristic rules follow our general pattern. Chapter 3 discusses how logics other than the intuitionistic one can be treated by introducing structural analogues to absurdity and negation. In addition to the proof-theoretic characterization of logical constants in Chapters 2 and 3, the framework developed in Chapter 1 allows us to define an extension of logic programming, which is the topic of Chapter 4. Here logic programs are regarded not merely as sets of definite Horn clauses (which are rules of levels 0 and 1), but as sets of arbitrary higher-level rules for atomic formulae. Using purely proof-theoretic methods, the basic theorems of soundness and completeness of SLD-resolution can be carried over to our more general framework. Chapter 5 contains a theory of relevance logic, which, if we distinguish between extensional and intensional rules, fits in very well with our approach to higher-level rules. This distinction generalizes ideas by Read (1984), Slaney (1984) and others, who admitted two binary structural operations (an extensional and an intensional one) for combining the antecedents of sequents. In the resulting framework logical constants can be characterized in the same uniform manner as before, but they now also include the standard constants of relevance logic, in particular relevant implication and relevant conjunction ("fusion"). It can be shown that our system, when restricted to the standard connectives of relevance logic, is equivalent to a very elegant formalism by Read and Slaney. Finally, in Chapter 6 we deal with Martin-Löf's logical theory, expanding it to include judgements of higher levels (which is a better term in this context than "rules of higher levels"). We confine ourselves to the basic principles of this system, considering it independently of its embedding in type theory. Martin-Löf's logical system is of particular philosophical interest because of its "presuppositional" nature: It does not allow us to even *assume* that something is true if we have not shown before that it is a proposition. As a consequence of this feature, the contents of higher-level judgements cannot be expressed by logically complex propositions in all cases. This distinguishes the approach from all frameworks considered in the preceding chapters.

Chapter 1

Structural Inference Schemata for Higher-Level Rules

In this chapter we present different possibilities of how to treat higher-level rules in a sequent-style framework. We first deal with rules which are not quantified and then extend our approach to rules in which generality can be expressed.

§ 1. Iterating the rule arrow

Suppose some notion of "formula" is given. (In the propositional case, no further specification is necessary.) *Rules of level n* are then defined as follows:

Each formula is a rule of level 0. If X is a nonempty finite set of rules of maximum level n and A is a formula, then $(X \Rightarrow A)$ is a rule of level $n+1$. $\emptyset \Rightarrow A$ is an abbreviation for A (and thus a rule of level 0). The elements of X are called the *premises* and A the *conclusion* of $(X \Rightarrow A)$.

We use A, B, C and D as syntactic variables for formulae, R and R' for rules, and U, V, W, X, Y and Z for finite sets of rules, where all of them can be primed or have indices. In the notation of rules, outer parentheses are normally omitted. Furthermore, set brackets surrounding premises of rules are left out, i.e., $R_1, \ldots, R_n \Rightarrow A$ is to be understood as $\{R_1, \ldots, R_n\} \Rightarrow A$. The identification of $\emptyset \Rightarrow A$ with A is both technically convenient and agrees with the intended meaning of a formula A when taken as a rule. It allows us to write the general form of a rule as

$$(X_1 \Rightarrow B_1), \ldots, (X_n \Rightarrow B_n) \Rightarrow A \ ,$$

including the case where one of the premises is just B_i or even where the rule is nothing but the formula A (without any premises).

We assume in the following that a set of rules is distinguished as the set of *primitive rules*. In many cases (e.g. in the case of logical rules), this set will be given by a finite set of rule schemata. However, the formulation of a structural framework is independent of which rules are considered primitive and how they are specified. It is the central idea of structural frameworks in our sense that the handling of rules is described in an abstract way, whatever their content may be.

The intended meaning of a higher-level rule, which is to be formally captured by certain structural inference schemata, can be stated as follows. A rule of level 0, i.e., a formula A, allows us to assert A. A rule of level 1, i.e., a rule of the form $B_1, \ldots, B_n \Rightarrow A$ allows us to pass over to A, provided all B_i $(1 \leq i \leq n)$ have already been proved. A rule of level greater than 1 of the form $(X_1 \Rightarrow B_1), \ldots, (X_n \Rightarrow B_n) \Rightarrow A$ allows us to pass over to A, provided for each i $(1 \leq i \leq n)$, B_i has been proved from X_i. Here X_i may itself contain higher-level rules, so that we have a system with rules both as primitive rules and as assumptions.

Another way of explaining the meaning of a rule, which is perhaps clearer, is to tell (1) how to *apply* and (2) how to *establish* a rule. A rule A of level 0 is applied by asserting A, and established by proving A. A rule $X \Rightarrow A$ is applied by passing over to A provided the rules in X have been established, and it is established by proving A where the rules in X may be applied as assumptions. Additional assumptions may be present, upon which rules to be established depend and which in rule applications are carried over from premises to conclusions.

We have used the term "prove" in an informal sense, applying it to statements made under assumptions (as in natural deduction). The term "derive" will be used only formally in connection with sequents. As discussed in the introduction, we express the idea of something's being proved or provable from assumptions by derivations in a sequent calculus. A *sequent* is an expression of the form

$$X \vdash R\ ,$$

where X is called the *antecedent* and R the *succedent.* A *restricted sequent* is of the form

$$X \vdash A\ ,$$

i.e., a sequent with only a formula allowed as succedent. The first formalism described in the following uses restricted sequents, all others use sequents. If we work with restricted sequents, a terminology like "$X \vdash Y \Rightarrow A$" is nevertheless allowed. In that case, it is considered a metalinguistic abbreviation for $X \cup Y \vdash A$. Concerning the antecedent of a sequent, we write as usual $X, Y \vdash R$ for $X \cup Y \vdash R$ and $X, R \vdash R'$ for $X \cup \{R\} \vdash R'$, etc. Furthermore, we use $R \dashv\vdash R'$ as an abbreviation for $\{R \vdash R', R' \vdash R\}$, and $X \vdash Y$ as an abbreviation for $\{X \vdash R_1, \ldots, X \vdash R_n\}$, if Y is $\{R_1, \ldots, R_n\}$, and for the empty set, if Y is empty. In this way, $X \vdash Y$ is also defined for systems with restricted sequents, since $X \vdash R$ is defined for such systems.

Let SC be any of the sequent calculi defined in the following. Then a *derivation* in SC is a tree of sequents (or restricted sequents, if applicable) which is constructed from *inferences* of the form

$$\frac{X_1 \vdash R_1 \ldots X_n \vdash R_n}{X \vdash R}\ .$$

(The specification "in SC" is, as usual, omitted when it is clear from the context which system is meant.) A derivation is also called a derivation *of* its lowermost sequent. The inferences allowed are given by the set of *inference schemata* of SC. The sequents above an inference line are called *upper sequents*, the sequent below the line *lower sequent* of this inference. This terminology also applies to inference schemata. The upper sequents of an inference or inference schema are considered to form a set, which means in particular that the order of branches in a derivation is not important. Nevertheless, since the upper sequents are given in a certain order when stating an inference schema, we can refer to the right or left upper sequent of an inference schema or of its application in a derivation.

If for each i $(1 \leq i \leq n)$, $\mathcal{D}_i$ is a derivation of $X_i \vdash R_i$, then the set $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ is called a derivation of the set $\{X_1 \vdash R_1, \ldots, X_n \vdash R_n\}$. As a limiting case the empty set is considered a derivation of the empty set. The length of a derivation is the number of inference steps used in it. A sequent $X \vdash R$ is called *derivable* in SC if there is a derivation in SC of $X \vdash R$. A set of sequents is called derivable if each element of the set is derivable. The empty set of sequents is always considered derivable.

An *inference schema* is called *admissible* in SC, if its lower sequent is derivable in SC, provided its upper sequents are derivable in SC. Two inference schemata I and I' are called *equivalent* in SC if I is admissible in SC enlarged with I', and I' is admissible in SC enlarged with I. Two systems SC and SC$'$ are called *equivalent* if each inference schema of SC is admissible in SC$'$ and vice versa. Obviously, this means the same as

$$X \vdash R \text{ is derivable in SC} \quad \text{iff} \quad X \vdash R \text{ is derivable in SC}',$$

so that equivalence of systems is an equivalence relation.

Furthermore, we define derivability and equivalence notions for rules with respect to a system SC with a fixed set of primitive rules. A *rule* R is called *derivable* in SC if the sequent $\emptyset \vdash R$ is derivable. Rules R and R' are called *equivalent* if $R \dashv\vdash R'$ is derivable. Finite sets of rules X and X' are called equivalent if $X \dashv\vdash X'$ is derivable. Rules R and R' are called *equivalent as primitive rules* if the following holds: R' is derivable, if R is added to the set of primitive rules, and R is derivable, if R' is added to the set of primitive rules. Sets X and X' are called *equivalent as sets of primitive rules* if each element of X' is derivable when X is added to the set of primitive rules, and each element of X is derivable when X' is added to the set of primitive rules.

In the following we describe four structural formalisms for the reasoning with higher level rules, called SC1 (with a variant SC1$'$), SC2, SC3 and SC4, which are proved to be equivalent. Table 1 at the end of this section gives a synopsis of their inference schemata. The underlying set of primitive rules is assumed to be fixed for all calculi.

Formalism 1 (SC1)

This formalism works with restricted sequents. It has only two inference schemata, one for the application of assumptions rules and one for the application of primitive rules:

$$\frac{X \vdash Y}{X, (Y \Rightarrow A) \vdash A} \text{(App-A)}$$

$$Y \Rightarrow A\,\frac{X \vdash Y}{X \vdash A}\,(\text{App-P}) \quad (\text{provided } Y \Rightarrow A \text{ is a primitive rule}).$$

From here on we mention primitive rules to the left of the inference line at which they "enter" a derivation. Note that according to our abbreviations, (App-A) and (App-P) include

$$\frac{}{X, A \vdash A}\,(\text{Ini})$$

and

$$A\,\frac{}{X \vdash A}\,(\text{Axiom})\ ,$$

respectively, as a special cases. If $\emptyset \Rightarrow A$ and A had been distinguished as rules, these two inference schemata would have to be added to the calculus, and $\emptyset \Rightarrow A \dashv\vdash A$ could be formally derived. Each branch of a derivation in SC1 must start with an inference of the form (Ini) or (Axiom). Since the set X in the antecedent of the lower sequent of such an inference can be arbitrarily augmented by a set Y without Y being affected by later applications of (App-A) and (App-P), the inference schema of thinning

$$\frac{X \vdash A}{X, Y \vdash A}\,(\text{Thin})$$

is admissible in SC1.

A variant of SC1, called SC1$'$, works with (unrestricted) sequents. Its inference schemata are those of SC1 (where upper sequents of the form $X \vdash Y \Rightarrow A$ are now understood as they stand and not as abbreviations for $X, Y \vdash A$), and in addition

$$\frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,(\Rightarrow +)\ .$$

Lemma 1.1 SC1 and SC1$'$ are equivalent, i.e., $X \vdash R$ is derivable in SC1 iff $X \vdash R$ is derivable in SC1$'$ (where in SC1$'$ $X \vdash R$ is used as an abbreviation if R is not a formula).

Proof A derivation of $X \vdash R$ in SC1 is obtained from a derivation of $X \vdash R$ in SC1$'$ by simply omitting all applications of $(\Rightarrow +)$. Conversely, given a derivation of $X \vdash R$ in SC1, we have to insert applications of $(\Rightarrow +)$ at all places where the abbreviatory reading of $U \vdash Z \Rightarrow B$ as $U, Z \vdash B$ in upper sequents of inference schemata is used. This gives us a derivation of $X, Y \vdash A$ in SC1$'$ if R is $Y \Rightarrow A$. Application of $(\Rightarrow +)$ then yields a derivation of $X \vdash R$. □

Lemma 1.2 The following inference schemata are admissible in SC1$'$:

$$\frac{}{R \vdash R}\,(\text{Refl}) \qquad\qquad R\,\frac{}{\emptyset \vdash R}\,(\text{Prim}) \quad (R \text{ primitive rule})$$

$$\frac{X \vdash R}{X, Y \vdash R}\,(\text{Thin}) \qquad\qquad \frac{X \vdash R \quad Y, R \vdash R'}{X, Y \vdash R'}\,(\text{Trans}')$$

$$\frac{X \vdash Y \Rightarrow A}{X, Y \vdash A}\,(\Rightarrow -)\ .$$

Proof (Thin) is admissible in SC1′ since it is admissible in SC1.
($\Rightarrow -$) is admissible, since ($\Rightarrow +$) is the only inference schema of SC1′ allowing to introduce the rule arrow "$\Rightarrow$" to the right of the turnstile.
(Refl) We use induction on the level of R. If R is a formula, $R \vdash R$ follows by (App-A). If R has the form $Y \Rightarrow A$, we know by induction hypothesis and (Thin) that $Y \vdash Y$ can be derived. By (App-A) we obtain $Y, (Y \Rightarrow A) \vdash A$, and by ($\Rightarrow +$) $R \vdash R$.
(Prim) Suppose R is of the form $Y \Rightarrow A$. Then we obtain $Y \vdash Y$ by (Refl) and (Thin), thus $Y \vdash A$ by (App-P) and $\emptyset \vdash Y \Rightarrow A$ by ($\Rightarrow +$).
(Trans) We use induction on the pair $\langle$level of R, length of derivations of $Y, R \vdash R'\rangle$. If the derivation of $Y, R \vdash R'$ is of the form

$$A\,\frac{}{Z \vdash A} \quad (A \text{ primitive rule})\ ,$$

then $X, Y \vdash R'$ is $X, Y \vdash A$ and can be obtained by the a step of the same kind. If it is of the form

$$\frac{}{Z, A \vdash A}\ ,$$

and R is in Z, then $X, Y \vdash R'$ is $U, A \vdash A$ for some U and can be obtained by a step of the same kind. If R is A, then $X \vdash R$ is $X \vdash A$, from which $X, Y \vdash R'$ is obtained by (Thin). Suppose the length of the derivation of $Y, R \vdash R'$ is greater than 1. If (App-P) or ($\Rightarrow +$) is applied in the last step, we pass to their upper sequents, use the induction hypothesis (with the left upper sequent $X \vdash R$ of (Trans′) being unchanged), and apply the step in question again. If a step of (App-A) of the form

$$\frac{U \vdash Z}{U, (Z \Rightarrow B) \vdash B}$$

is used to derive the right upper sequent of (Trans′), and R is in U, we proceed in the same way. If R is $Z \Rightarrow B$, then we obtain from the left upper sequent of (Trans′) $X, Z \vdash B$ by ($\Rightarrow -$). Each rule in Z is of lower level than R. Hence from $U \vdash Z$ and $X, Z \vdash B$ we obtain by iterated application of the induction hypothesis $X, U \vdash B$, which is the same as $X, Y \vdash R'$. □[1]

[1] Avron (1986) investigated a system with the inference schemata (App-A) (but restricted to nonempty Y), (Thin) and

$$\frac{}{A \vdash A}\,(\text{I})\ .$$

Formalism 2 (SC2)

The intuitive idea behind the inference schemata (App-A) and (App-P) was that *by applying* the rule $Y \Rightarrow A$ we may pass over from Y to A. If $Y \Rightarrow A$ is not a primitive rule, it is added to the set of assumptions. The idea underlying SC2 is that a rule must first be *stated* before it can be applied. As all subsequent formalisms, SC2 uses (unrestricted) sequents. Its inference schemata are:

$$\frac{}{R \vdash R}\,(\text{Refl}) \qquad R\,\frac{}{\emptyset \vdash R}\,(\text{Prim}) \quad (R \text{ primitive rule})$$

$$\frac{X \vdash R}{X, Y \vdash R}\,(\text{Thin}) \qquad \frac{X \vdash Y \quad X \vdash Y \Rightarrow A}{X \vdash A}\,(\text{App})$$

$$\frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,(\Rightarrow +)\ .$$

Theorem 1.3 SC1′ and SC2 are equivalent.

Proof We first show that all inference schemata of SC1′ are admissible in SC2: Applications of (App-A) or (App-P) with Y empty can be replaced by applications of (Refl) or (Prim), respectively, in SC2, followed by (Thin). Applications of (App-A) with Y nonempty can be replaced by the following derivation in SC2:

$$\frac{\dfrac{X \vdash Y}{X, (Y \Rightarrow A) \vdash Y}\,(\text{Thin}) \qquad \dfrac{\dfrac{}{Y \Rightarrow A \vdash Y \Rightarrow A}\,(\text{Refl})}{X, (Y \Rightarrow A) \vdash Y \Rightarrow A}\,(\text{Thin})}{X, (Y \Rightarrow A) \vdash A}\,(\text{App})\ .$$

Applications of (App-P) with Y nonempty can be replaced by the following derivation in SC2:

$$\frac{X \vdash Y \qquad \dfrac{\dfrac{}{\emptyset \vdash Y \Rightarrow A}\,(\text{Prim})}{X \vdash Y \Rightarrow A}\,(\text{Thin})}{X \vdash A}\,(\text{App})\ .$$

(These three schemata are exactly as strong as our (App-A) with Y allowed to be empty.) Instead of using one schema like (App-P) for the application of primitive rules, Avron added for each primitive rule $Y \Rightarrow A$ the inference schema

$$\frac{X \vdash Y}{X \vdash A}\ .$$

Although this is conceptually different from our approach, derivations in Avron's system and derivations in SC1 can easily be translated into each other. (Actually, they are identical if in Avron's system (Thin) is dropped and (I) is replaced by (Ini).) Avron also demonstrates that $X \vdash A$ is derivable in his system if A can be proved from X in the natural deduction system of Schroeder-Heister (1984a).

The schema $(\Rightarrow +)$ is itself an inference schema of both SC1′ and SC2. For the converse direction, we only have to show that (App) is admissible in SC1′. The rest follows from Lemma 1.2 and the fact that $(\Rightarrow +)$ is an inference schema of SC1′. Concerning (App), we observe that $(\Rightarrow -)$ is admissible in SC1′ (Lemma 1.2). Therefore the admissibility of (App) can be reduced to that of (Trans′) in SC1′ (Lemma 1.2). □

Formalism 3 (SC3)

SC3 is just a slight modification of SC2. As inference schemata we take (Refl), (Prim), (Thin) and $(\Rightarrow +)$ as before, but replace (App) by

$$\frac{X \vdash R \quad R \vdash R'}{X \vdash R'} \text{(Trans)}$$

and add

$$\frac{X \vdash Y \Rightarrow A}{X, Y \vdash A} (\Rightarrow -) \ .$$

The schema $(\Rightarrow -)$ must be added since it is no longer admissible when (App) is replaced by (Trans). (Trans) may look strange, since it does not seem to incorporate the idea of the application of assumption rules, which is central to our approach. However, (Trans) is obviously equivalent to

$$\frac{X \vdash Y \quad X, Y \vdash A}{X \vdash A}$$

(one simply does several steps in one). Like (App) the latter schema now expresses the idea of rule application, the only difference being that the rule $Y \Rightarrow A$ is not *explicitly* stated in the succedent of the right upper sequent, but *implicitly* with its premises in the antecedent.

Theorem 1.4 SC2 and SC3 are equivalent.

Proof By Lemma 1.2, all inference schemata of SC3 are admissible in SC1′, (Trans) being a special case of (Trans′). Therefore, by Theorem 1.3, they are admissible in SC2. For the converse direction we only have to show that (App) is admissible in SC3. Due to $(\Rightarrow +)$ and $(\Rightarrow -)$, this is reduced to applications of (Trans). □

Formalism 4 (SC4)

This formalism uses an inference schema that introduces the rule arrow to the left of the turnstile. It was proposed by Hallnäs (1986) as a most convenient system for the treatment of logic programming with higher-level rules. SC4 has the following inference schemata:

$$\frac{}{X, A \vdash A} \text{(Ini)} \qquad\qquad Y \Rightarrow A \, \frac{X \vdash Y}{X \vdash A} \text{(App-P)} \quad (Y \Rightarrow A \text{ primitive rule})$$

$$\frac{X \vdash Y \quad X, A \vdash R}{X, (Y \Rightarrow A) \vdash R}\,(\Rightarrow\vdash) \qquad\qquad \frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,(\Rightarrow +)\ .$$

Obviously, $(\Rightarrow +)$ and $(\Rightarrow\vdash)$ correspond to the introduction rules for implication to the right and left of the turnstile in "symmetric" sequent calculi (see Gentzen 1935a). The schema (Ini) is not a limiting case of $(\Rightarrow\vdash)$, whereas in SC1 and SC1′ (to which SC4 is related by its schema (App-P)), it was a special case of (App-A). Thus it must be formulated as a separate inference schema.

Theorem 1.5 SC1′ and SC4 are equivalent.

Proof The inference schema (App-A) is admissible in SC4, since for empty Y applications of (App-A) can be replaced by applications of (Ini) and for nonempty Y by the following derivation:

$$\frac{X \vdash Y \quad \dfrac{}{X, A \vdash A}\,(\text{Ini})}{X, (Y \Rightarrow A) \vdash A}\,(\Rightarrow\vdash)\ .$$

To show that $(\Rightarrow\vdash)$ is admissible in SC1′, we apply (App-A) to the left upper sequent of $(\Rightarrow\vdash)$, yielding $X, (Y \Rightarrow A) \vdash A$, and use the admissibility of (Thin) and (Trans′) in SC1′ (Lemma 1.2). (Ini) is a special case of (App-A). □

For computational purposes, by which we here mean backward reasoning, the systems SC2 and SC3 are not appropriate, since certain information is lost in the lower sequents of (App) and (Trans) which is present in the upper sequents.

Table 1 presents the inference schemata of the sequent calculi defined. From here on, a double line is used to represent two schemata, one read downwards and one read upwards. The label to the right indefinitely refers to both schemata. The downward schema is specifically referred to by adding a "+" to this label, the upward schema by adding a "−".

§ 2. Rules with rules as conclusions

Rules are formed by iteration of the rule arrow "⇒" to the left, i.e. they have the form $X \Rightarrow A$ for a finite set of rules X. Philosophically, it would make good sense if we also allowed for rules to occur to the right of the rule arrow. Such rules have the general form $X \Rightarrow R$, where R may itself be of the form $Y \Rightarrow R'$. A rule $X \Rightarrow R$ would enable us to establish R, if we have established X. These extended rules will be called "r-rules" (where "r" is to remind one of "right iteration"), as distinguished from rules simpliciter. In the structural framework for relevance logic (see Ch. 5 below) rules with rules as conclusions will become the standard form of rules. The concept of an r-rule is defined as follows:

Each formula is an r-rule of level 0. If X is a nonempty finite set of r-rules of maximum level m and R an r-rule of level n, then $(X \Rightarrow R)$ is an r-rule of level $\max(m, n) + 1$. $\emptyset \Rightarrow R$ is considered an abbreviation for R.

TABLE 1

Inference schemata of the structural systems SC1 to SC4

(Rules mentioned to the left of an inference line are assumed to be primitive.)

SC1

$$\frac{X \vdash Y}{X, (Y \Rightarrow A) \vdash A}\,(\text{App-A}) \qquad Y \Rightarrow A\,\frac{X \vdash Y}{X \vdash A}\,(\text{App-P})$$

SC1′

$$\frac{X \vdash Y}{X, (Y \Rightarrow A) \vdash A}\,(\text{App-A}) \qquad Y \Rightarrow A\,\frac{X \vdash Y}{X \vdash A}\,(\text{App-P}) \qquad \frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,({\Rightarrow}+)$$

SC2

$$\frac{}{R \vdash R}\,(\text{Refl}) \qquad R\,\frac{}{\emptyset \vdash R}\,(\text{Prim}) \qquad \frac{X \vdash R}{X, Y \vdash R}\,(\text{Thin})$$

$$\frac{X \vdash Y \quad X \vdash Y \Rightarrow A}{X \vdash A}\,(\text{App}) \qquad \frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,({\Rightarrow}+)$$

SC2+ (see § 2) has the letter "R" instead of "A" in (App) and $({\Rightarrow}+)$

SC3

$$\frac{}{R \vdash R}\,(\text{Refl}) \qquad R\,\frac{}{\emptyset \vdash R}\,(\text{Prim}) \qquad \frac{X \vdash R}{X, Y \vdash R}\,(\text{Thin})$$

$$\frac{X \vdash Y \quad R \vdash R'}{X \vdash R'}\,(\text{Trans}) \qquad \genfrac{}{}{1.5pt}{}{X, Y \vdash Y}{X \vdash Y \Rightarrow A}\,({\Rightarrow})$$

SC3+ (see § 2) has the letter "R" instead of "A" in (App) and $({\Rightarrow}+)$.

SC4

$$\frac{}{X, A \vdash A}\,(\text{Ini}) \qquad Y \Rightarrow A\,\frac{X \vdash Y}{X \vdash A}\,(\text{App-P})$$

$$\frac{X \vdash Y \quad X, A \vdash R}{X, (Y \Rightarrow A) \vdash R}\,({\Rightarrow}\vdash) \qquad \frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,({\Rightarrow}+)$$

All notational conventions of the previous section extend to the present case, the syntactic variables for rules and sets of rules now also comprising r-rules and sets of r-rules. As a codification of the reasoning with r-rules we present the following modification of SC3, called SC3+:

$$\frac{}{R \vdash R}\,(\text{Refl}) \qquad R\,\frac{}{\emptyset \vdash R}\,(\text{Prim}) \quad (R \text{ primitive rule})$$

$$\frac{X \vdash R}{X, Y \vdash R}\,(\text{Thin}) \qquad \frac{X \vdash R \quad R \vdash R'}{X \vdash R'}\,(\text{Trans}) \qquad \frac{X, Y \vdash R}{\overline{\overline{X \vdash Y \Rightarrow R}}}\,(\Rightarrow)\ .$$

In $(\Rightarrow)$ we have the syntactic variable "R" for rules instead of "A" for formulae. Otherwise the inference schemata of SC3 and SC3+ are literally the same. But, of course, there is a difference in the interpretation of the syntactic variables which in SC3+ refer to r-rules and sets thereof.

Since every rule is an r-rule, every derivation in SC3 is a derivation in SC3+, provided the primitive rules of SC3 are primitive rules of SC3+. Therefore SC3+ is at least as strong as SC3. We shall show that SC3+ can be embedded in SC3, so that it does not properly extend SC3.

We define a transformation s which translates r-rules and finite sets of r-rules into rules and finite sets of rules in a natural way.

$s(A) = A$
$s(Y \Rightarrow R) = (s(Y), X) \Rightarrow A,$ if $s(R) = X \Rightarrow A$
$s(\emptyset) = \emptyset$
$s(\{R_1, \ldots, R_n\}) = \{s(R_1), \ldots, (R_n)\}.$

By induction on the level of r-rules R it can be shown that $s(R)$ is always defined and is a rule. The second clause in this definition can be rephrased as

$$s(Y_1 \Rightarrow (Y_2 \Rightarrow \ldots \Rightarrow (Y_n \Rightarrow A)\ldots)) = (s(Y_1), \ldots, s(Y_n)) \Rightarrow A\ .$$

Lemma 1.6 Let X and Y be finite sets of r-rules and R be an r-rule. Then the pair of inference schemata

$$\frac{s(X), s(Y) \vdash s(R)}{\overline{\overline{s(X) \vdash s(Y \Rightarrow R)}}}$$

is admissible in SC3.

Proof If $s(R)$ is of the form $Z \Rightarrow A$, then the upper sequent is of the form $s(X), s(Y) \vdash Z \Rightarrow A$ and the lower sequent of the form $s(X) \vdash (s(Y), Z) \Rightarrow A$. The assertion then follows by use of $(\Rightarrow)$. □

Lemma 1.7 Let R be an r-rule. Then $R \dashv\vdash s(R)$ is derivable in SC3+.

Proof by induction on the level of R. If R is a formula A, the assertion is trivial. Suppose R is of the form $Y \Rightarrow R'$ for nonempty Y. Then we have the following derivation in SC3+:

$$\dfrac{\dfrac{}{R \vdash R}\,(\text{Refl})}{R, Y \vdash R'}\,(\Rightarrow)$$

$$\vdots \qquad \text{induction hypothesis with respect to } Y \text{ and } R', \text{ (Trans) and } (\Rightarrow)$$

$$\dfrac{\dfrac{R, s(Y) \vdash s(R')}{R, s(Y), X \vdash A}\,(\Rightarrow), s(R') \text{ supposed to be of the form } X \Rightarrow A}{R \vdash s(R)}\,(\Rightarrow)\ .$$

The sequent $s(R) \vdash R$ is derived similarly by reading this derivation upwards, with the R to the left of the turnstile replaced by $s(R)$. □

Theorem 1.8 (Embedding of SC3+ in SC3) Suppose a set of r-rules is given as the set of primitive r-rules for SC3+. Let for each such primitive r-rule R' the rule $s(R')$ be a primitive rule for SC3. Then for any r-rule R and finite set of r-rules X, the sequent $X \vdash R$ is derivable in SC3+ iff $s(X) \vdash s(R)$ is derivable in SC3.

Proof Suppose a derivation of $X \vdash R$ is SC3+ is given. Applications of the inference schemata (Refl), (Prim), (Thin) and (Trans) immediately become applications of the corresponding inference schemata of SC3 when r-rules and sets thereof are transformed by s. For applications of $(\Rightarrow)$ we use Lemma 1.6. Conversely, suppose a derivation of $s(X) \vdash s(R)$ in SC3 is given. We replace each step

$$s(R)\,\dfrac{}{\emptyset \vdash s(R)}\,(\text{Prim})$$

by the following steps in SC3+:

$$\dfrac{R\,\dfrac{}{\emptyset \vdash R}\,(\text{Prim}) \qquad \dfrac{}{R \vdash s(R)}\,\text{Lemma 1.7}}{\emptyset \vdash s(R)}\,(\text{Trans})\ ,$$

and leave all other steps unchanged. The result is a derivation of $s(X) \vdash s(R)$ in SC3+. By using Lemma 1.7, (Trans) and $(\Rightarrow)$ we obtain a derivation of $X \vdash R$ in SC3+. □

These results show that it is justified to consider r-rules of the form $X \Rightarrow R$ to be abbreviations for rules $s(X \Rightarrow R)$ and to work with SC3 rather than SC3+. This is no longer possible in systems which use more complicated assumption structures than just sets to the left of the turnstile. The structural framework for relevance logic discussed in Ch. 5 will serve as an example. It will be formulated as a strongly modified version of SC3+.

For the other sequent calculi discussed in the previous section, versions for r-rules are available, too. We here only mention SC2+, since it will be the starting point for our formulation of Martin-Löf's logic in Ch. 6. The system SC2+ results from SC2 by replacing the syntactic variable "A" by "R":

$$\frac{}{R \vdash R}\,(\text{Refl}) \qquad R\,\frac{}{\emptyset \vdash R}\,(\text{Prim}) \quad (R \text{ primitive rule})$$

$$\frac{X \vdash R}{X, Y \vdash R}\,(\text{Thin}) \qquad \frac{X \vdash Y \quad X \vdash Y \Rightarrow R}{X \vdash R}\,(\text{App})$$

$$\frac{X, Y \vdash R}{X \vdash Y \Rightarrow R}\,(\Rightarrow +)\ .$$

Corresponding to Theorem 1.8, SC2+ can be embedded in SC2: It is easy to see that SC2+ and SC3+ are equivalent. Since we also know that SC2 and SC3 are equivalent (Theorem 1.4), the assertion is a consequence of Theorem 1.8.

§ 3. Rules with generality

In this section we extend the formalism of § 3 by rules which express generality. Following the practice of many proof-theoretic investigations, we distinguish free and bound variables by different kinds of symbols. We suppose that infinitely many free variables and infinitely many bound variables are at our disposal and that *terms* are defined so as to include free variables, but not bound variables. Here variables and terms are understood in the sense of first-order logic, i.e., as *individual* variables and *individual* terms. It may be mentioned, however, that our approach can in principle be carried over to higher-order logic. We use a, b, c as syntactic symbols for free variables, x, y, z for bound variables, t for terms, $\underline{a}$, $\underline{b}$, $\underline{c}$ for (possibly empty) finite sequences of *distinct* free variables, $\underline{x}$, $\underline{y}$, $\underline{z}$ for (possibly empty) finite sequences of *distinct* bound variables, and $\underline{t}$ for (possibly empty) finite sequences of (not necessarily distinct) terms.

We assume that formulae are defined in such a way that bound variables only occur in the scope of a variable-binding operator and that bound variables are separated in the following sense: If in the scope of a variable-binding operator $Q_1\underline{x}$ a variable-binding operator $Q_2\underline{y}$ occurs (where Q_2 may be the same operator as Q_1), then $\underline{x}$ and $\underline{y}$ are disjoint. As before, A, B, C are syntactic symbols for formulae.

If $\underline{a}$ is $a_1, \ldots, a_n$, $\underline{t}$ is $t_1, \ldots, t_n$ and E is an arbitrary expression, then $E\,[\underline{a}/\underline{t}]$ denotes the result of simultaneously substituting all a_i in E by t_i $(1 \leq i \leq n)$. $E\,[\underline{a}/\underline{x}]$ is defined similarly. The substitution operation is of strongest precedence, i.e., it always applies to the smallest syntactic unit to its left. When writing $[\underline{a}/\underline{t}]$ or $[\underline{a}/\underline{x}]$ we always asume that $\underline{a}$ and $\underline{x}$ or $\underline{a}$ and $\underline{t}$, respectively, are of the same length. If $\underline{a}$ and $\underline{t}$ are empty, then $E\,[\underline{a}/\underline{t}]$ is E. Of the concept of "formula" we require that it be closed under substitution, i.e., $A[\underline{a}/\underline{t}]$ is a formula for each formula A.

To express generality in rules we use the prefix $\Rightarrow_{\underline{x}}$ for arbitrary nonempty $\underline{x}$, called the *generality operator*. Rules with generality are then defined by adding the following clause to the inductive definition of rules of higher levels:

If R is a rule of level n, which does not start with the generality operator, and if $\underline{a}$ and $\underline{x}$ are nonempty, and no element of $\underline{x}$ occurs in R, then $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ is a rule of level $n+1$. (Note that according to our conventions, in $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ the substitution $[\underline{a}/\underline{x}]$ applies to R and not to $\Rightarrow_{\underline{x}} R$.) $\Rightarrow_{\underline{x}} R$ is identified with R, if $\underline{x}$ is empty.

This definition treats the generality operator as a variable-binding operator, and the above restriction for formulae (bound variables only occur in the scope of variable-binding operators and are separated) are carried over to rules. We do *not* allow for rules of the form $\Rightarrow_{\underline{x}}(\Rightarrow_{\underline{y}} R\,[\underline{a}/\underline{y}])[\underline{b}/\underline{x}]$. Instead of $\Rightarrow_{\underline{x}}(X\,[\underline{a}/\underline{x}] \Rightarrow A\,[\underline{a}/\underline{x}])$ we also write $X\,[\underline{a}/\underline{x}] \Rightarrow_{\underline{x}} A\,[\underline{a}/\underline{x}]$. However, it should be clear that we strictly distinguish the generality of a rule from the rule arrow as expressing hypothetical dependence. This is different from our natural deduction style presentation in Schroeder-Heister (1984b) as well as from Martin-Löf's treatment of "hypothetico-general" judgements (Martin-Löf 1985a), where generality is only treated in connection with hypothetical dependence.

The intended meaning of a rule $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ can be stated by telling how to apply and how to establish it (possibly under assumptions). To *apply* $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ means to apply $R\,[\underline{a}/\underline{t}]$ for an arbitrary $\underline{t}$. In other words, the rule $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ may be used as if it were any of its instances $R\,[\underline{a}/\underline{x}]$. To *establish* $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ means to prove R schematically, i.e., to prove $R\,[\underline{a}/\underline{b}]$ for some $\underline{b}$ whose elements do not occur in the assumptions under which $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ is established.

The generality operators within the premises of a rule may be considered to express eigenvariable conditions: Described in natural deduction style, a rule of the form

$$((X_1\,[\underline{a}_1/\underline{x}_1] \Rightarrow_{\underline{x}_1} A_1\,[\underline{a}_1/\underline{x}_1]), \ldots, (X_n\,[\underline{a}_n/\underline{x}_n] \Rightarrow_{\underline{x}_n} A_n\,[\underline{a}_n/\underline{x}_n])) \Rightarrow A$$

allows us to pass over to A, if for each i $(1 \leq i \leq n)$, A_i has been proved from X_i in such a way that the elements of $\underline{a}_i$ do not occur in assumptions on which A_i depends except X_i.

When working with restricted sequents, we use $X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ as an abbreviation for $X \vdash R\,[\underline{a}/\underline{b}]$, where $\underline{b}$ is chosen in a unique way such that no element of $\underline{b}$ occurs in X. If R is of the form $Y \Rightarrow A$, this means $X, Y\,[\underline{a}/\underline{b}] \vdash A\,[\underline{a}/\underline{b}]$.

Before we describe which schemata for generality are to be added to the different formalisms, we must explain the relation between assumption rules and primitive rules, which is somewhat different in the present case. If we can derive the sequent $X \vdash R$ in one of our formalisms, we want to be able to also derive $X\,[\underline{a}/\underline{t}] \vdash R\,[\underline{a}/\underline{t}]$. Rules taken as assumptions are not problematic in that respect since they occur somewhere within X, the free variables $\underline{a}$ in these rules being substituted by $\underline{t}$. Primitive rules, however, are so to speak "applied from outside", i.e. substitution in the sequent to be derived does not affect them. Therefore we have two possibilities: (1) We require that the set of primitive rules be closed under substitution, i.e., if R is a primitive rule then so is $R\,[\underline{a}/\underline{t}]$. Primitive rules are then used in a derivation as they stand. If we need an instance $R\,[\underline{a}/\underline{t}]$ of a primitive rule R in a derivation, we just use this $R\,[\underline{a}/\underline{t}]$ as a primitive rule of its own. (2) Formation of substitution instances $R\,[\underline{a}/\underline{t}]$ of primitive rules R is not a closure condition for the set of primitive rules, but a step performed when using primitive rules in a derivation. We shall follow the second alternative.

Furthermore, we assume that primitive rules are of the form $Y \Rightarrow A$, i.e. they do not start with a generality operator (but may contain generality operators within Y). Otherwise we would have to introduce inference schemata which allow to specialize a primitive rule of the form $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ to $R\,[\underline{a}/\underline{t}]$. This specialization is already contained in our inference schemata, when R is taken as a primitive rule. Intuitively, the free variables of a primitive rule must be understood as generalized.

The sequent calculi SC1, SC1′, SC2, SC3 and SC4 are to be altered as follows, when generality in rules is added. The resulting systems are called SC1Q, SC1′Q, SC2Q, SC3Q and SC4Q, where the "Q" should remind one of "quantified". When the notation $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ is used, it is always assumed that R does not start with a generality operator.

SC1Q

The inference schema (App-P) is replaced by

$$Y \Rightarrow A\,\frac{X \vdash Y\,[\underline{a}/\underline{t}]}{X \vdash A\,[\underline{a}/\underline{t}]}\,(\text{App-P}*)\quad (Y \Rightarrow A \text{ primitive rule})$$

and the inference schema

$$\frac{X, R\,[\underline{a}/\underline{t}] \vdash A}{X, \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}] \vdash A}\,(\text{Gen}\vdash)$$

is added to SC1. (App-P∗) expresses the idea that one applies a primitive rule by applying any of its instances, and (Gen⊢) corresponds to the idea that to apply $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ as an assumption rule means to apply any of its special cases $R\,[\underline{a}/\underline{t}]$.

SC1′Q

As SC1Q this system has (App-P∗) as an inference schema. It uses a version of (Gen⊢) with rules in the succedent:

$$\frac{X, R\,[\underline{a}/\underline{t}] \vdash R'}{X, \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}] \vdash R'}\,(\text{Gen}\vdash)\ .$$

Furthermore, the inference schema

$$\frac{X \vdash R}{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}\,(\vdash\text{Gen})\quad (\text{provided no element of } \underline{a} \text{ occurs in } X)$$

is added. The relationship between the inference schemata (Gen⊢) and (⊢Gen) and Gentzen's schemata for the universal quantifier being introduced to the right and left of the turnstile (Gentzen 1935a) is obvious.

Lemma 1.9 If $X \vdash R$ is derivable in SC1′Q, then so is $X\,[\underline{a}/\underline{t}] \vdash R\,[\underline{a}/\underline{t}]$.

Proof by induction on the length of derivations. If an inference schema different from $(\vdash\text{Gen})$ is applied in the last step, the assertion follows by a straightforward application of the induction hypothesis. If $(\vdash\text{Gen})$ is applied in the last step in the following form:

$$\frac{X \vdash R'}{X \vdash \Rightarrow_{\underline{x}} R' \, [\underline{b}/\underline{x}]} \; ,$$

then by using the induction hypothesis we obtain a derivation of $X \vdash R' \, [\underline{b}/\underline{c}]$ for $\underline{c}$ disjoint with $\underline{a}$ and no element of $\underline{c}$ occurring in X or in $\underline{t}$. Again by induction hypothesis we obtain a derivation of $X \, [\underline{a}/\underline{t}] \vdash R' \, [\underline{b}/\underline{c}][\underline{a}/\underline{t}]$, hence by $(\vdash\text{Gen})$ one of $X \, [\underline{a}/\underline{t}] \vdash \Rightarrow_{\underline{x}} R' \, [\underline{b}/\underline{c}][\underline{a}/\underline{t}][\underline{c}/\underline{x}]$, which is the same as $X \, [\underline{a}/\underline{t}] \vdash \Rightarrow_{\underline{x}} R' \, [\underline{b}/\underline{x}][\underline{a}/\underline{t}]$. □

Lemma 1.10 The inference schemata mentioned in Lemma 1.2 as well as

$$R \, \frac{}{\emptyset \vdash R \, [\underline{a}/\underline{t}]} \, (\text{Prim}*) \quad (R \text{ primitive rule})$$

and

$$\frac{X \vdash \Rightarrow_{\underline{x}} R \, [\underline{a}/\underline{x}]}{X \vdash R \, [\underline{a}/\underline{t}]} \, (\text{Spec})$$

are admissible in SC1′Q.

Proof (Thin) Suppose $X \vdash R$ is derivable in SC1′Q. We want to show that $X, Y \vdash R$ is derivable. Let $\underline{a}$ contain all free variables of Y, and let $\underline{b}$ be of the same length as $\underline{a}$, but without free variables occurring in the derivation of $X \vdash R$. Then by adding $Y \, [\underline{a}/\underline{b}]$ to the antecedent of each sequent in the derivation, we obtain a derivation of $X, Y \, [\underline{a}/\underline{b}] \vdash R$. By Lemma 1.9 we obtain a derivation of $X, Y \vdash R$, substituting $\underline{b}$ by $\underline{a}$.
($\Rightarrow -$) As in the proof of Lemma 1.2.
(Spec) Since $(\vdash\text{Gen})$ is the only inference schema which allows to introduce the generality operator to the right of the turnstile, its converse is admissible. Then (Spec) follows by Lemma 1.9.
(Refl) In addition to the argument in the proof of Lemma 1.2, we must show that if $R \vdash R$ is derivable, then so is $\Rightarrow_{\underline{x}} R \, [\underline{a}/\underline{x}] \vdash \Rightarrow_{\underline{x}} R \, [\underline{a}/\underline{x}]$. This follows by application of $(\text{Gen}\vdash)$ and $(\vdash\text{Gen})$.
(Prim$*$) (which includes (Prim)): By (Refl) and (Thin) we know that $Y \, [\underline{a}/\underline{t}] \vdash Y \, [\underline{a}/\underline{t}]$, if R is $Y \Rightarrow A$. By application of (App-P$*$) we obtain $Y \, [\underline{a}/\underline{t}] \vdash A \, [\underline{a}/\underline{t}]$, hence $\emptyset \vdash R \, [\underline{a}/\underline{t}]$ by $(\Rightarrow +)$.
(Trans) In addition to the argument in the proof of Lemma 1.2 we must consider the case where (Trans′) is used with its right upper sequent $Y, R \vdash R'$ being derived by using $(\text{Gen}\vdash)$ in the last step, where R is $\Rightarrow_{\underline{x}} R_1 \, [\underline{a}/\underline{x}]$. Then we have a derivation of $Y, R_1 \, [\underline{a}/\underline{t}] \vdash R'$ which is shorter than that of $Y, R \vdash R'$. From the left upper sequent of (Trans′), which is of the form $X \vdash \Rightarrow_{\underline{x}} R_1 \, [\underline{a}/\underline{x}]$, we obtain $X \vdash R_1 \, [\underline{a}/\underline{t}]$ by (Spec). Application of the induction hypothesis gives us the assertion. □

A *variant* R' of a rule R is the result of renaming bound variables in R in such a way that different occurrences of a bound variable x in R are transformed into different occurrences of a bound variable y in R' (where y may be identical with x), and occurrences of different bound variables x_1 and x_2 in R are transformed into occurrences of different bound variables y_1 and y_2 in R'. R is considered a variant of itself. X and X' are called variants of each other if for each R in X there is a variant R' of R such that R' is in X', and for each R' in X' there is a variant R of R' such that R is in X.

Lemma 1.11 If for all *formulae* A and A' which are variants of each other $A \dashv\vdash A'$ is derivable in SC1′Q, then for all *rules* R and R' which are variants of each other $R \dashv\vdash R'$ is derivable in SC1′Q.

Proof by induction on the level of R. If R is a formula, the assertion holds by hypothesis. If R is of the form $X \Rightarrow B$, then R' is of the form $X' \Rightarrow B'$ such that X' and X as well as B and B' are variants of each other. By induction hypothesis and (Thin) (Lemma 1.10), $X \dashv\vdash X'$ and $B \dashv\vdash B'$ are derivable. By (App-A) we obtain $X, (X' \Rightarrow B') \vdash B'$ and $X', (X \Rightarrow B) \vdash B$, and by (Thin), (Trans), $(\Rightarrow +)$ and $(\Rightarrow -)$ (Lemma 1.10) we obtain $R \dashv\vdash R'$. If R is of the form $\Rightarrow_{\underline{x}} R_1[\underline{a}/\underline{x}]$, then R' is of the form $\Rightarrow_{\underline{y}} R_1' [\underline{a}/\underline{y}]$ where R_1 and R_1' are variants of each other (here $\underline{x}$ and $\underline{y}$ may be identical). By induction hypothesis, $R_1 \dashv\vdash R_1'$ is derivable, therefore by (Gen$\vdash$) and ($\vdash$Gen) also $R \dashv\vdash R'$. □

Structural subformulae (in short: s-subformulae) of rules are defined as follows: Each formula is an s-subformula of itself. Each s-subformula of an element of X is an s-subformula of X. A is an s-subformula of $X \Rightarrow A$. Each s-subformula of R is an s-subformula of $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$.

We do not speak of subformulae (simpliciter), since this term should be reserved when the internal components of logically complex formulae are considered, and not only the formula components of rules. Since $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ is the same as $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{b}][\underline{b}/\underline{x}]$ for $\underline{b}$ not containing free variables of R, $A\,[\underline{a}/\underline{b}]$ is an s-subformula of $\Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]$ if A is one. According to our inductive definition of s-subformulae on the structure of rules, an s-subformula A of R is always related to certain places in the formation tree of R, although not A but only some expression $A\,[\underline{a}/\underline{x}]$ actually appears in R. Therefore it makes sense to speak of an *occurrence* of an s-subformula A in R, which is considered as the pair consisting of A and a place in the formation tree of R. We say that such an occurrence is *replaced* by B if the formation of R is repeated by using B at this place instead of A.

Theorem 1.12 (Replacement theorem) Let R be a rule in which A occurs at certain places as an s-subformula. Let R' result from R by replacing these occurrences of A in R by A'. If $A \dashv\vdash A'$ is derivable in SC1′Q, then so is $R \dashv\vdash R'$.

Proof by induction on the level of R. If R is a formula, the assertion holds by hypothesis. If R is of the form $X \Rightarrow B$, then R' is of the form $X' \Rightarrow B'$ such that X' and B' result from X and B, respectively, by replacing certain occurrences of A by A'. By induction hypothesis and (Thin) (Lemma 1.10), $X \dashv\vdash X'$ and $B \dashv\vdash B'$ are derivable. By (App-A) we obtain $X, (X' \Rightarrow B') \vdash B'$ and $X', (X \Rightarrow B) \vdash B$, and by (Thin), (Trans), $(\Rightarrow +)$ and $(\Rightarrow -)$ (Lemma 1.10) we obtain $R \dashv\vdash R'$. If R is of the form $\Rightarrow_{\underline{x}} R_1 [\underline{a}/\underline{x}]$, then R' is of the form $\Rightarrow_{\underline{x}} R'_1 [\underline{a}/\underline{x}]$ where R_1 results from R'_1 by replacing certain occurrences of A by A'. By induction hypothesis, $R_1 \dashv\vdash R'_1$ is derivable, therefore by (Gen$\vdash$) and ($\vdash$Gen) also $R \dashv\vdash R'$. (This proof is parallel to that of Lemma 1.11.) □

SC2Q and SC3Q

(Prim) is replaced by

$$R \frac{}{\emptyset \vdash R\,[\underline{a}/\underline{t}]}\,(\text{Prim}*) \quad (R \text{ primitive rule})$$

and the inference schema ($\vdash$Gen) as well as

$$\frac{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}{X \vdash R\,[\underline{a}/\underline{t}]}\,(\text{Spec})\ ,$$

is added.

SC4Q

The alterations and additions are the same as with SC1′, i.e., (App-P) is replaced by (App-P$*$), and (Gen$\vdash$) and ($\vdash$Gen) are added.

Theorem 1.13 The formalisms SC1Q, SC1′Q, SC2Q, SC3Q and SC4Q are equivalent.

Proof The equivalence of SC1Q and SC1′Q is proved as in Lemma 1.1, with ($\vdash$Gen) being treated parallel to $(\Rightarrow +)$. For the equivalence of SC1′Q and SC2Q the proof of Theorem 1.3 can be taken over, with Lemma 1.11 now playing the role of Lemma 1.2. Furthermore, (App-P$*$) and (Prim$*$) must replace (App-P) and (Prim), respectively. Similarly, Theorems 1.4 and 1.5 can be adopted for the equivalence of SC2Q, SC3Q and SC4Q. □

Table 2 presents those inference schemata of the different systems which are specific for generality.

TABLE 2

Inference schemata of the structural systems with generality

(Rules to the left of an inference line are assumed to be primitive. Inference schemata which are not specific for generality are not repeated here—their labels refer to Table 1.)

SC1Q

(App-A) $$Y \Rightarrow A \, \frac{X \vdash Y\,[\underline{a}/\underline{t}]}{X \vdash A\,[\underline{a}/\underline{t}]}\,(\text{App-P}*) \qquad \frac{X, R\,[\underline{a}/\underline{t}] \vdash A'}{X, \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}] \vdash A'}\,(\text{Gen}\vdash)$$

SC1′Q

(App-A)
(⇒ +) $$Y \Rightarrow A \, \frac{X \vdash Y\,[\underline{a}/\underline{t}]}{X \vdash A\,[\underline{a}/\underline{t}]}\,(\text{App-P}*) \qquad \frac{X, R\,[\underline{a}/\underline{t}] \vdash R'}{X, \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}] \vdash R'}\,(\text{Gen}\vdash)$$

$$\frac{X \vdash R}{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}\,(\vdash\text{Gen})$$ (provided no element of $\underline{a}$ occurs in X)

SC2Q

(Refl)
(Thin) $$R \, \frac{}{\emptyset \vdash R\,[\underline{a}/\underline{t}]}\,(\text{Prim}*) \qquad \frac{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}{X \vdash R\,[\underline{a}/\underline{t}]}\,(\text{Spec})$$

(App)
(⇒ +) $$\frac{X \vdash R}{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}\,(\vdash\text{Gen})$$ (provided no element of $\underline{a}$ occurs in X)

SC3Q

(Refl)
(Thin) $$R \, \frac{}{\emptyset \vdash R\,[\underline{a}/\underline{t}]}\,(\text{Prim}*) \qquad \frac{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}{X \vdash R\,[\underline{a}/\underline{t}]}\,(\text{Spec})$$

(Trans)
(⇒) $$\frac{X \vdash R}{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}\,(\vdash\text{Gen})$$ (provided no element of $\underline{a}$ occurs in X)

SC4Q

(Ini)
(⇒ ⊢) $$Y \Rightarrow A \, \frac{X \vdash Y\,[\underline{a}/\underline{t}]}{X \vdash A\,[\underline{a}/\underline{t}]}\,(\text{App-P}*) \qquad \frac{X, R\,[\underline{a}/\underline{t}] \vdash R'}{X, \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}] \vdash R'}\,(\text{Gen}\vdash)$$

(⇒ +) $$\frac{X \vdash R}{X \vdash \Rightarrow_{\underline{x}} R\,[\underline{a}/\underline{x}]}\,(\vdash\text{Gen})$$ (provided no element of $\underline{a}$ occurs in X)

Chapter 2

Higher-Level Rules for Logical Constants

Logical constants of propositional logic were treated in a natural deduction context in Schroeder-Heister (1981, 1984a). Here we shall deal with the general case which includes both propositional constants and quantifiers. Logical constants in this general sense were the topic of Schroeder-Heister (1984b), but within a conceptually different framework. Whereas in that paper quantifiers were considered variable-binding operators, they will now be treated as operators having predicate terms as arguments. A predicate term is either a primitive predicate or obtained by use of λ-abstraction from a formula. This way of proceeding is both technically and intuitively more perspicuous. Technically, the problems of variable-binding (such as renaming of bound variables etc.) are concentrated on the λ-operator and taken away from the variety of logical constants. Intuitively, Frege's convincing analysis of quantifiers as second-order predicates of concepts (see e.g. Frege 1893) seems to us to be better represented by connecting variable-binding with concept formation than with quantifier application.

In § 1 we specify our language with logical constants and λ-abstraction, § 2 presents primitive rules for logical constants of arbitrary type, and § 3 relates them to the standard connectives of intuitionistic logic and proves their (functional) completeness. Since the completeness problem is treated in the papers mentioned above, and, for the propositional case, also in von Kutschera (1968), Zucker and Tragesser (1978) and Prawitz (1979/1982), we do not spell out all proofs in detail.

§ 1. Predicate terms and logical constants

We specify our language more precisely than was necessary in the previous chapter. By an *arity* we understand a natural number. By a *type* we understand a tuple $\langle k_1, \ldots, k_n\rangle$ of arities, including the empty tuple $\langle\rangle$ as a limiting case. Let arbitrarily many *primitive predicates*, each with an associated arity, and arbitrarily many *logical constants*, each with an associated type be given. Logical constants are syntactically denoted by S (sometimes primed and with indices). Let free and bound variables and terms be given as in

Ch. 1 § 3, the same notational conventions being in force. Then *formulae* and *predicate terms*[1] (where formulae are 0-ary predicate terms) are defined as follows.

Each n-ary primitive predicate is an n-ary predicate term. For each n-ary predicate term T, $T(t_1, \dots, t_n)$ is a formula (in short: $T(\underline{t})$, if $\underline{t}$ is $t_1, \dots, t_n$). (This is to include the case $n = 0$, in which T itself is a formula.)
If A is a formula, $\underline{x}$ is of length n, and no element of $\underline{x}$ occurs in A, then $\lambda\underline{x}\, A\, [\underline{a}/\underline{x}]$ is an n-ary predicate term. (According to our conventions, the substitution $[\underline{a}/\underline{x}]$ is understood with respect to A and not with respect to $\lambda\underline{x}A$.)
If $T_1, \dots, T_n$ are predicate terms of arities $k_1, \dots, k_n$, respectively, and S is of type $\langle k_1, \dots, k_n\rangle$, then $S(T_1, \dots, T_n)$ is a formula.

We use T as a syntactic variable for predicate terms, $\underline{T}$ for finite sequences of predicate terms and $\underline{A}$ for finite sequences of formulae (all syntactic variables may be primed or have indices). Therefore a formula starting wih a logical constant can be written as $S(\underline{T})$. A logical constant S of type $\langle k_1, \dots, k_n\rangle$ such that $k_i = 0$ for all i $(1 \le i \le k)$, is also called a k-ary *propositional connective*. A formula starting with a propositional connective can be written as $S(\underline{A})$. Rules with generality are defined as in Ch. 1 § 3.

For λ-abstraction the law of λ-*conversion* is supposed to hold. Therefore we take all rules of the following form as primitive rules:

$$\begin{array}{l} (\lambda\underline{x}\, A\, [\underline{a}/\underline{t}])(\underline{t}) \Rightarrow A\, [\underline{a}/\underline{t}] \\ A\, [\underline{a}/\underline{t}] \Rightarrow (\lambda\underline{x}\, A\, [\underline{a}/\underline{t}])(\underline{t}) \ , \end{array} \qquad (\lambda)$$

provided $\underline{a}$, $\underline{x}$ and $\underline{t}$ are of the same length and no element of $\underline{x}$ occurs in A.

If T is a predicate term of the form $\lambda\underline{x}\, A\, [\underline{a}/\underline{x}]$, then A is called an *immediate subformula* of T. The *subformulae* of a formula A are A itself, the subformulae of immediate subformulae of T if A is $T(\underline{t})$, and the subformulae of immediate subformulae of $T_1, \dots, T_n$, if A is $S(T_1, \dots, T_n)$. As in the case of s-subformulae of a rule, it makes sense to speak of *occurrences* of subformulae in a formula A.

For the characterization of logical constants we need specific notions of formula schemata and rule schemata. Whereas in the case of inference schemata and in (λ) we simply used our syntactic variables to express a schematic reading, in the case of logical constants it is convenient to have rule schemata as formal objects, since we want to speak about them in a uniform manner. For that purpose we introduce additional symbols $p_1, p_2, p_3, \dots$, called *schematic letters*. A *rule schema* of type $\langle k_1, \dots, k_n\rangle$ is then defined as formed like a rule, in which the schematic letters $p_1, \dots, p_n$ may occur as if they were primitive predicates of arities $k_1, \dots, k_n$, respectively, but in which no "genuine" primitive predicates and no terms except free variables are allowed to occur. If a rule schema of type $\langle k_1, \dots, k_n\rangle$ does not contain a rule arrow or a generality operator, it is also called a *formula schema* of type $\langle k_1, \dots, k_n\rangle$.

Intuitively, the only "content" which may be present in rule schemata is given by logical constants. All specific "content" which may be due to primitive predicates, primitive function symbols or individual constants is excluded.

[1]The expression "predicate term" is taken from Prawitz (1965, Ch. VI).

For example, $p_1 \Rightarrow p_2$ is a rule schema of type $\langle 0,0\rangle$ (but also e.g. of type $\langle 0,0,n\rangle$ for any n), and $\Rightarrow_{\underline{y}} S_1(\lambda x S_2(p_2(x,y),p_3(x)))$ is a rule schema of type $\langle n,2,1\rangle$ for any n, if S_2 is of type $\langle 0,0\rangle$ and S_1 of type $\langle 1\rangle$. The schematic letters p_1, p_2 are not letters whose arity is fixed once and for all as is the case with primitive predicates. In other words, two rule schemata may be rule schemata of different types, though they contain the same schematic letters $p_1,\ldots,p_n$. But as soon as a type $\langle k_1,\ldots,k_n\rangle$ is fixed for a rule schema, the arity of p_i $(1 \le i \le n)$ is fixed as well.

If a type $\langle k_1,\ldots,k_n\rangle$ is given and $\underline{p}$ is $p_1,\ldots,p_n$, then formula schemata are denoted by $\alpha[\underline{p}]$ or simply α, rule schemata by $\varrho[\underline{p}]$ or simply ϱ and finite sets of rule schemata (of the same type) by $\Delta[\underline{p}]$ or simply Δ (sometimes primed or with indices). *Variants* of formula and rule schemata are explained in the same way as for formulae and rules. If $\underline{T}$ is a sequence $T_1,\ldots,T_n$ of predicate terms of arities $k_1,\ldots,k_n$, respectively, then $\varrho[\underline{T}]$ results from $\varrho[\underline{p}]$ by first passing over to a variant of $\varrho[\underline{p}]$ containing no bound variables which also occur in $\underline{T}$, and then replacing p_i by T_i for all i $(1 \le i \le n)$. Obviously, if $\varrho[\underline{p}]$ is a rule schema, then $\varrho[\underline{T}]$ is a rule, called an *instance* of $\varrho[\underline{p}]$.

Two rule schemata $\varrho_1[\underline{p}]$ and $\varrho_2[\underline{p}]$ of the same type are called *schematically equivalent*, if $\varrho_1[\underline{p}] \dashv\vdash \varrho_2[\underline{p}]$ can be derived by using the schematic letters $p_1,\ldots,p_n$ as if they were primitive predicates. If $\varrho_1[\underline{p}]$ and $\varrho_2[\underline{p}]$ are schematically equivalent, then $\varrho_1[\underline{T}] \dashv\vdash \varrho_2[\underline{T}]$ for all appropriate $\underline{T}$, provided it holds that $R_1 \dashv\vdash R_2$ is derivable for all R_1 and R_2 which are variants of each other (see Lemma 2.2 below). (We just have to take an appropriate variant $\underline{T}'$ of $\underline{T}$ such that no bound variables of $\underline{T}'$ occur in the derivation of $\varrho_1[\underline{p}] \dashv\vdash \varrho_2[\underline{p}]$, replace $\underline{p}$ by $\underline{T}'$ in this derivation throughout, and then apply Lemma 1.11 to the result.) Two rule schemata ϱ_1 and ϱ_2 are called *equivalent* (simpliciter), if for each instance R of ϱ_1 there is a set X of instances of ϱ_2 such that $X \vdash R$ is derivable and conversely with ϱ_1 and ϱ_2 interchanged. Two sets of rule schemata Δ_1 and Δ_2 are called equivalent, if for each instance R of a rule schema in Δ_1 there is a set X of instances of rule schemata in Δ_2 such that $X \vdash R$ is derivable and conversely with Δ_1 and Δ_2 interchanged. Two rule schemata ϱ_1 and ϱ_2 are called *equivalent as schemata for primitive rules* if for each instance R of ϱ_1, $\emptyset \vdash R$ is derivable after adding the instances of ϱ_2 as primitive rules, and conversely with ϱ_1 and ϱ_2 interchanged. Two sets of rule schemata Δ_1 and Δ_2 are called equivalent as sets of schemata for primitive rules if for each instance R of a rule schema in Δ_1, $\emptyset \vdash R$ is derivable after adding the instances of Δ_2 as primitive rules, and conversely with Δ_1 and Δ_2 interchanged. Obviously, if Δ_1 and Δ_2 are equivalent, then they are equivalent as schemata for primitive rules, but not necessarily vice versa.

§ 2. Primitive rules for logical constants

The basic idea underlying our proof-theoretic characterization of logical constants is that with a complex formula certain conditions are associated from which it can be inferred via introduction rules. Corresponding elimination rules state that everything that can be inferred from the premises of each of the introduction rules can be inferred from their conclusion. They guarantee that the conclusion of the introduction rules has no less content than have the possible premises when taken together. The conditions associated with a logical constant are called "assertibility conditions", following a terminology proposed by Dummett (e.g. 1977, Ch. 7). However, the issues of justifying introduction and elimination rules for logical constants within a theory of meaning like Dummett's will be discussed elsewhere. Furthermore, we shall not discuss notions of validity which give a semantics to logical constants in terms of proofs (see Prawitz 1973, 1974, 1985, Schroeder-Heister 1983, 1985), or concepts of uniqueness for logical constants (see Došen and Schroeder-Heister 1985, 1987).

Let S be a logical constant of type $\langle k_1, \ldots, k_n \rangle$. An *assertibility condition* for S is a finite set $\Delta[\underline{p}]$ of rule schemata of type $\langle k_1, \ldots, k_n \rangle$, where $\underline{p}$ is $p_1, \ldots, p_n$. We assume that with each logical constant S a finite set $\mathbb{D}(S)$ of assertibility conditions for S is associated, i.e., $\mathbb{D}(S)$ is either empty or

$$\mathbb{D}(S) = \{\Delta_1[\underline{p}], \ldots, \Delta_m[\underline{p}]\}$$

for $m \geq 1$. The sets $\mathbb{D}(S)$ for the logical constants must fulfil the following requirement: All logical constants available in the language considered can be ordered in a sequence

$$S_1, S_2, S_3, \ldots, S_i, \ldots$$

in such a way that for every i, $\mathbb{D}(S_i)$ contains no logical constants S_j for $j \geq i$. In other words: Logical constants are given assertibility conditions step by step. The assertibility conditions for a logical constant may only refer to logical constants for which assertibility conditions have already been given.

Let S be of type $\langle k_1, \ldots, k_n \rangle$. Let $\underline{p}$ be $p_1, \ldots, p_n$. Let in the following $\underline{T}$ always stand for $T_1, \ldots, T_n$, such that each T_i is k_i-ary $(1 \leq i \leq n)$. Then the set of rule schemata of *S-introduction* is empty, if $\mathbb{D}(S)$ is empty, and consists of

$$\begin{array}{l} \Delta_1[\underline{p}] \Rightarrow S(\underline{p}) \\ \quad\vdots \\ \Delta_m[\underline{p}] \Rightarrow S(\underline{p}) \end{array} \qquad (S\text{-I})$$

if $\mathbb{D}(S) = \{\Delta_1[\underline{p}], \ldots, \Delta_m[\underline{p}]\}$. An S-introduction rule is an instance of a schema of S-introduction. The rule schema of *S-elimination* is of type $\langle k_1, \ldots, k_n, 0 \rangle$, since it contains in addition to $\underline{p}$ a schematic letter p_{n+1} to be instantiated by formulae:

$$(S(\underline{p}), (\Delta_1[\underline{p}][\underline{a}_1/\underline{x}_1] \Rightarrow_{\underline{x}_1} p_{n+1}), \ldots, (\Delta_m[\underline{p}][\underline{a}_m/\underline{x}_m] \Rightarrow_{\underline{x}_m} p_{n+1})) \Rightarrow p_{n+1} (S\text{-E})$$

Here for each i $(1 \leq i \leq m)$, $\underline{a}_i$ contains *all* free variables occurring in $\Delta_i[\underline{p}]$ and $\underline{x}_i$ is chosen in such a way that the substitution of $\underline{a}_i$ by $\underline{x}_i$ is defined. If $\mathbb{D}(S)$ is empty, this schema reduces to

$$S(\underline{p}) \Rightarrow p_{n+1}$$

as a limiting case. An S-elimination rule is an instance of the schema of S-elimination.[2]

We now assume that the system considered contains the rules of λ-conversion (λ) and introduction and elimination rules for each logical constant as its primitive rules. It may be based on any of our structural frameworks with generality.

Lemma 2.1 If for each i $(1 \leq i \leq m)$, $\Delta_i(\underline{T}) \dashv\vdash \Delta_i(\underline{T}')$ is derivable, then so is $S(\underline{T}) \dashv\vdash S(\underline{T}')$.

Proof We use SC3Q. By using (S-E), we can derive

$$(S(\underline{T}), (\Delta_1(\underline{T})[\underline{a}_1/\underline{x}_1] \Rightarrow_{\underline{x}_1} S(\underline{T}'), \ldots, (\Delta_m(\underline{T})[\underline{a}_m/\underline{x}_m] \Rightarrow_{\underline{x}_m} S(\underline{T}')) \vdash S(\underline{T}') ,$$

and by using $\Delta_i(\underline{T}) \dashv\vdash \Delta_i(\underline{T}')$ and (Trans), we obtain

$$(S(\underline{T}), (\Delta_1(\underline{T}')[\underline{a}_1/\underline{x}_1] \Rightarrow_{\underline{x}_1} S(\underline{T}'), \ldots, (\Delta_m(\underline{T}')[\underline{a}_m/\underline{x}_m] \Rightarrow_{\underline{x}_m} S(\underline{T}')) \vdash S(\underline{T}') .$$

Then by (S-I), ($\vdash$Gen) and (Trans), $S(\underline{T}) \vdash S(\underline{T}')$ can be derived. The derivation of $S(\underline{T}') \vdash S(\underline{T})$ is symmetric to the one of $S(\underline{T}) \vdash S(\underline{T}')$. □

The rank of a formula is defined as follows:

rk$(A) = 0$, if A starts with a primitive predicate.
rk$((\lambda\underline{x}\, A\, [\underline{a}/\underline{x}])(\underline{t})) = \text{rk}(A)[\underline{a}/\underline{t}] + 1$.
rk$(S(\underline{T})) = \max(\{\text{rk}(A) : A \text{ is an } s\text{-subformula of } \Delta_i(\underline{T}) \text{ for some } i\ (1 \leq i \leq m)\}) + 1$

Lemma 2.2 If R and R' are variants of each other, then $R \dashv\vdash R'$ is derivable.

Proof Because of Lemma 1.11 we only have to show the assertion for formulae A and A'. We use induction on the rank of A. If A contains no bound variable, nothing is to show. If A is of the form $(\lambda\underline{x}\, A\, [\underline{a}/\underline{x}])(\underline{t})$, then A' is of the form $(\lambda\underline{y}\, A'\, [\underline{a}/\underline{y}])(\underline{t})$ where B' is a variant of B. By using λ-conversion (λ) and the induction hypothesis, we obtain the assertion. If A is $S(\underline{T})$, then A' is $S(\underline{T}')$, and for each i $(1 \leq i \leq m)$, $\Delta_i(\underline{T})$ and $\Delta_i(\underline{T}')$ differ with respect to certain s-subformulae which are variants of each other. Since these s-subformulae are of lower rank than A, by applying the induction hypothesis and Lemma 1.11 we obtain that $\Delta_i(\underline{T}) \dashv\vdash \Delta_i(\underline{T}')$ is derivable. Then by Lemma 2.1 we obtain the assertion. □

Theorem 2.3 Suppose B occurs in A at certain places as a subformula, and A' results from A by replacing these occurrences of B by B'. If $B \dashv\vdash B'$ derivable, then so is $A \dashv\vdash A'$.

[2]Obviously, these schemata are much simpler than those presented in Schroeder-Heister (1984b). This is not only due to the use of λ-abstraction, but also to the use of different symbols for free and bound variables. In particular, no restriction on the possible substitutes of p_{n+1} has to be required.

Proof By induction on the rank of A. If A is of the form $(\lambda\underline{x}\, C\,[\underline{a}/\underline{x}])(\underline{t})$, then we use λ-conversion (λ) and apply the induction hypothesis. If A is $S(\underline{T})$, then A' is $S(\underline{T'})$, and B occurs at certain places in s-subformulae of rules in $\Delta_i(\underline{T})$ $(1 \leq i \leq m)$. These s-subformulae are of lower rank than A, so that we can apply the induction hypothesis to them. By Theorem 1.12 we obtain that for all i $(1 \leq i \leq m)$, $\Delta_i(\underline{T}) \dashv\vdash \Delta_i(\underline{T'})$ is derivable. The assertion follows by Lemma 2.1. □

The intuition behind the schemata for introduction and elimination rules was that a logically compound formula somehow expresses the content of certain assertibility conditions. This can be made more precise by formally defining a notion of "common content". Let the *common content* of finite sets $X_1, \ldots, X_m$ of rules with respect to a finite sequence of free variables $\underline{a}$ be the set of all rules R such that for all $\underline{t}$ and for all i $(1 \leq i \leq m)$, $X_i[\underline{a}/\underline{t}] \vdash R$ is derivable (where, as a limiting case, $m = 0$ is allowed). Let the *content* of a formula A be the set of all rules such that $A \vdash R$ is derivable.[3] Now suppose $\mathbb{D}(S)$ is as above. Let $\underline{a}$ contain all free variables occurring in $\mathbb{D}(S)$.

Theorem 2.4 Let a calculus be given, whose primitive rules contain (λ). Then the S-introduction and S-elimination rules are derivable iff for all $\underline{T}$, the content of $S(\underline{T})$ is exactly the common content of $\Delta_1[\underline{T}], \ldots, \Delta_m[\underline{T}]$ with respect to $\underline{a}$.

Proof

(i) Suppose the S-introduction and S-elimination rules are derivable. If $S(\underline{T}) \vdash R$ is derivable, then so is $\Delta_i(\underline{T})[\underline{a}/\underline{t}] \vdash R$ for each i $(1 \leq i \leq m)$ and all $\underline{t}$ by (S-I). If $\Delta_i(\underline{T})[\underline{a}/\underline{t}] \vdash R$ is derivable for each i $(1 \leq i \leq m)$ and all $\underline{t}$, then $\Delta_i(\underline{T})[\underline{a}/\underline{t}] \vdash R$ is derivable for some $\underline{b}$ whose elements do not occur in R. Then $S(\underline{T}) \vdash R$ can be derived by using (S-E).

(ii) Suppose the content of $S(\underline{T})$ is the common content of $\Delta_1[\underline{T}], \ldots, \Delta_m[\underline{T}]$ with respect to $\underline{a}$. Since $S(\underline{T})$ itself is in the content of $S(\underline{T})$, $\Delta_i(\underline{T})[\underline{a}/\underline{t}] \vdash S(\underline{T})$ is derivable for each i $(1 \leq i \leq m)$, i.e., the S-introduction rules are derivable. Since

$$\Delta_i(\underline{T})[\underline{a}/\underline{t}], (\Delta_i(\underline{T})[\underline{a}_i/\underline{x}_i] \Rightarrow_{\underline{x}_i} A) \vdash A$$

is derivable for each i $(1 \leq i \leq m)$ and each $\underline{t}$, if $\underline{a}_i$ contains all free variables of $\Delta_i(\underline{p})$), the rule

$$((\Delta_1(\underline{T})[\underline{a}_1/\underline{x}_1] \Rightarrow_{\underline{x}_1} A), \ldots, (\Delta_m(\underline{T})[\underline{a}_m/\underline{x}_m] \Rightarrow_{\underline{x}_m} A)) \Rightarrow A$$

is in the common content of $\Delta_1(\underline{T}), \ldots, \Delta_m(\underline{T})$ with respect to $\underline{a}$. Therefore it is in the content of $S(\underline{T})$, which implies the derivability of the S-elimination rules. □

[3]Unlike in our informal expositions, "content" is here understood in a well-defined formal sense, which resembles Tarski's notion of content (see e.g. Tarski 1930).

Since the S-introduction and S-elimination rules are derivable in a system in which they are primitive rules, we have as a corollary that in our logical systems the content of $S(\underline{T})$ is always the common content of $\Delta_1[\underline{T}], \ldots, \Delta_m[\underline{T}]$ with respect to $\underline{a}$.

Our systematics of introduction and elimination rules allows us to prove a normalization theorem in the sense of Prawitz (1965) in very general terms, when treated within a structural formalism like SC1Q (which is very near to natural deduction formulations). This is carried out in detail for the propositional case in Schroeder-Heister (1981, 1982). It can be extended to all logical constants. Here we only mention without proof a corollary of this result. A logical constant S' is called an *immediate predecessor* of S, if S' occurs in $\mathbb{D}(S)$. S' is called a *predecessor* of S if S' is S or an immediate predecessor of a predecessor of S.

Theorem 2.5 (Separation of logical constants) If $X \vdash R$ is derivable, then there is a derivation of $X \vdash R$ in which only logical constants, which are predecessors of logical constants in $X \vdash R$, occur. (This means in particular that only introduction and elimination rules for predecessors of logical constants occurring in $X \vdash R$ are used.) □

If we only consider constants of propositional logic the situation becomes much easier. We describe this special case, since we shall refer to it later. Constants of propositional logic are of type $\langle 0, \ldots, 0 \rangle$, so that schematic letters $p_1, p_2, \ldots$ exclusively stand for formulae. An n-ary formula schema or rule schema is now defined as follows: Every schematic letter p_i $(1 \leq i \leq n)$ is an n-ary formula schema. If S is an k-ary logical constant and $\alpha_1, \ldots, \alpha_k$ are n-ary formula schemata, then $S(\alpha_1, \ldots, \alpha_k)$ is an n-ary formula schema. Each formula schema is a rule schema. If Δ is a nonempty finite set of n-ary rule schemata and α is an n-ary formula schema, then $\Delta \Rightarrow \alpha$ is an n-ary rule schema.

The introduction schemata for n-ary connectives are literally the same as before, except that the $\Delta_i[\underline{p}]$ are now understood as n-ary rule schemata in the sense just defined. In the elimination schema the generality operators can be omitted so that it takes the form

$$(S(\underline{p}), (\Delta_1[\underline{p}] \Rightarrow p_{n+1}), \ldots, (\Delta_m[\underline{p}] \Rightarrow p_{n+1})) \Rightarrow p_{n+1} \ .$$

In the "common content"-motivation the reference to free variables and terms can be omitted. This means that the S-introduction and S-elimination rules are derivable iff for all $\underline{A}$, the content of $S(\underline{A})$ is always the common content of $\Delta_1[\underline{A}], \ldots, \Delta_m[\underline{A}]$.

§ 3. The standard constants of intuitionistic logic and their completeness

Our uniform pattern for introduction and elimination schemata contains the introduction and elimination schemata for the standard constants &, $\vee$, $\supset$, $\perp$, $\forall$ and $\exists$ of intuitionistic logic as limiting cases. Table 3 presents for each of these constants type, assertibility conditions and introduction and elimination schemata. Binary connectives are written in their usual infix notation. The quantifiers $\forall$ and $\exists$ are no longer treated as variable-binding operators but as constants with unary predicate terms as arguments. E.g., $\forall x P(x)$

and $\exists x P(x)$ (in the usual notation) become $\forall(\lambda x P(x))$ and $\exists(\lambda x P(x))$. Furthermore, the schematic letters p_1, p_2 and p_3 are replaced by p, q and r, respectively.

TABLE 3
Standard constants of intuitionistic logic and their primitive rules

(Binary connectives are written in infix notation. The letters p, q and r stand for p_1, p_2 and p_3, respectively.)

S	Type of S	$\mathbb{D}(S)$	Schemata for I rules	Schemata for E rules
$\&$	$\langle 0,0\rangle$	$\{\{p,q\}\}$	$p,q \Rightarrow p\,\&\,q$	$(p\,\&\,q, (\{p,q\} \Rightarrow r)) \Rightarrow r$ equivalent: $p\,\&\,q \Rightarrow p$ $p\,\&\,q \Rightarrow q$
$\vee$	$\langle 0,0\rangle$	$\{\{p\},\{q\}\}$	$p \Rightarrow p \vee q$ $q \Rightarrow p \vee q$	$(p \vee q, (p \Rightarrow r), (q \Rightarrow r)) \Rightarrow r$
$\supset$	$\langle 0,0\rangle$	$\{\{p \Rightarrow q\}\}$	$(p \Rightarrow q) \Rightarrow p \supset q$	$(p \supset q, ((p \Rightarrow q) \Rightarrow r)) \Rightarrow r$ equivalent: $p \supset q, p \Rightarrow q$
$\bot$	$\langle\rangle$	$\emptyset$	missing	$\bot \Rightarrow p$
$\forall$	$\langle 1\rangle$	$\{\{\Rightarrow_x p(x)\}\}$	$\Rightarrow_x p(x) \Rightarrow \forall(p)$	$(\forall(p), (\Rightarrow_x p(x) \Rightarrow q)) \Rightarrow q$ equivalent as schema for primitive rules: $\forall(p) \Rightarrow p(a)$
$\exists$	$\langle 1\rangle$	$\{\{p(a)\}\}$	$p(a) \Rightarrow \exists(p)$	$(\exists(p), (p(x) \Rightarrow_x q)) \Rightarrow q$

In the case of &, $\supset$ and $\forall$, the elimination schemata required by our uniform pattern differ from what one would expect. However, rule schemata which are more usual are equivalent to them (at least as schemata for primitive rules). As an example, we prove the equivalence of

$$(\forall(p), (\Rightarrow_x p(x) \Rightarrow q)) \Rightarrow q \qquad (\forall\text{-E})$$

and

$$\forall(p) \Rightarrow p(a) \qquad (\forall\text{-E}')$$

as schemata for primitive rules in the structural calculus SC3Q: Let T be a unary predicate term. Then $\emptyset \vdash \Rightarrow_x T(x) \Rightarrow T(a)$ can be derived by using (Spec). By using (Trans) we obtain

$$(\forall(T), (\Rightarrow_x T(x) \Rightarrow T(a))) \Rightarrow T(a) \vdash \forall(T) \Rightarrow T(a)\ .$$

Using (Prim$*$) for the instance of $\forall$-E with T substituted for p and $T(a)$ for q, then with ($\Rightarrow$) and (Trans) we get $\emptyset \vdash \forall(T) \Rightarrow T(a)$. Conversely, using (Prim$*$) for the instance of $\forall$-E$'$ with T substituted for p we obtain $\emptyset \vdash \forall(T) \Rightarrow T(a)$, hence, by ($\Rightarrow -$) and ($\vdash$Gen), $\forall(T) \vdash \Rightarrow_x T(x)$. Since $\Rightarrow_x T(x), (\Rightarrow_x T(x) \Rightarrow A) \vdash A$ can be derived by using (Refl) and ($\Rightarrow -$), we obtain by (Trans) and ($\Rightarrow +$): $\emptyset \vdash (\forall(T), (\Rightarrow_x T(x) \Rightarrow A)) \Rightarrow A$.

Because of λ-conversion and Theorem 2.3, the instances of ($\forall$-E$'$) can be written as

$$\forall(\lambda x A[a/x]) \Rightarrow A\ ,$$

which looks more common. (Note that the instantiation of the free variable a in A to some term t is a matter of the structural schemata (App-P$*$) or (Prim$*$), not a matter of the formulation of the primitive rule.)

Negation can be defined by using absurdity $\bot$: $\mathbb{D}(\neg) = \{\{p \Rightarrow \bot\}\}$. Schemata for introduction and elimination rules are

$$(p \Rightarrow \bot) \Rightarrow \neg p \qquad (\neg\text{-I})$$

and

$$\begin{array}{l}(\neg p, (p \Rightarrow \bot) \Rightarrow q) \Rightarrow q \\ (\text{or equivalently: } \neg p, p \Rightarrow q)\ .\end{array} \qquad (\neg\text{-E})$$

A constant $\top$ of type $\langle\rangle$ expressing truth could be defined by $\mathbb{D}(\top) = \emptyset$. Its only introduction rule would be

$$\top\ , \qquad (\top\text{-I})$$

and the schema for elimination rules would be

$$\top, p \Rightarrow p\ . \qquad (\top\text{-E})$$

Obviously the $\top$-elimination rules are derivable. In the present context, $\top$ is definable by $\bot \supset \bot$. In other contexts, however, a constant like $\top$ may play an independent role (see Ch. 5 on relevance logic).

The standard intuitionistic constants described in Table 3 are complete in the sense that they suffice to define every other constant that has introduction and elimination rules which follow our general pattern. We call this completeness "functional completeness", since it reminds one of the well-known functional completeness of certain sets of connectives in classical propositional logic. One must be aware, however, that logical constants in our sense are not understood as truth functions. We only give a sketch of how to prove this functional completeness.

Let a type for rule schemata be fixed. We use $\forall \underline{x}\varrho[\underline{a}/\underline{x}]$ as an abbreviation for $\forall(\lambda x_1 \ldots \forall(\lambda x_n(\varrho[\underline{a}/\underline{x}]))\ldots)$, if $\underline{x}$ is $x_1,\ldots,x_n$. If $\underline{a}$ and $\underline{x}$ are empty, $\forall \underline{x}\varrho[\underline{a}/\underline{x}]$ is just ϱ. The analogous convention holds for $\exists$. We first define a translation g of rule schemata and sets of rule schemata into formula schemata as follows:

$g(\alpha) = \alpha$
$g(\{\varrho_1,\ldots,\varrho_n\}) = g(\varrho_1) \mathbin{\&} \ldots \mathbin{\&} g(\varrho_n)$
$g(\emptyset) = \bot \supset \bot$
$g(\Delta \Rightarrow \alpha) = g(\Delta) \supset g(\alpha)$
$g(\Rightarrow_{\underline{x}} \varrho[\underline{a}/\underline{x}]) = \forall \underline{x}(g(\varrho)[\underline{a}/\underline{x}])$.

Roughly speaking, the comma is translated by conjunction, the empty set by the formula $\bot\supset\bot$ expressing truth, the rule arrow by implication and the generality operator by the universal quantifier. It can then be shown that Δ and $g(\Delta)$ are schematically equivalent, and that for the derivation of $\Delta \dashv\vdash g(\Delta)$ only introduction and elimination rules for &, $\supset$, $\forall$ and $\bot$ are needed. Now we define a translation h of "non-standard" logical constants:

$h(S(\underline{p})) = \exists \underline{x}_1(g(\Delta_1[\underline{p}])[\underline{a}_1/\underline{x}_1]) \vee \ldots \vee \exists \underline{x}_m(g(\Delta_m[\underline{p}])[\underline{a}_m/\underline{x}_m])$, if $\mathbb{D}(S) = \{\Delta_1[\underline{p}],\ldots,\Delta_m[\underline{p}]\}$ and for each i $(1 \leq i \leq m), \underline{a}_i$ contains all free variables of $\Delta_i[\underline{p}]$ (and therefore of $g(\Delta_i[\underline{p}])$), and no element of $\underline{x}_i$ occurs in $\Delta_i[\underline{p}]$
$h(S(\underline{p})) = \bot$ if $\mathbb{D}(S) = \emptyset$.

It is easy to see that $h(p \mathbin{\&} q) = p \mathbin{\&} q$, $h(p \vee q) = p \vee q$, $h(p \supset q) = p \supset q, h(\bot) = \bot$, $h(\forall(p)) = \forall x p(x)$, $h(\exists(p)) = \exists x p(x)$. In other words, the standard propositional connectives are literally translated into themselves, and the standard quantifiers are translated into something that is the same modulo λ-conversion. Each other constant S is translated into a formula schema which contains no logical constants except the standard constants and constants occurring in $\mathbb{D}(S)$. Roughly speaking, by h each assertibility condition is first translated via g, then its free variables are understood existentially quantified (since they only occur in the *premises* of the corresponding introduction rules), and different assertibility conditions are connected disjunctively (since they represent alternatives for the introduction of S). It can then be shown by using only primitive rules for S and for the standard constants that $S(\underline{p})$ and $h(S(\underline{p}))$ are schematically equivalent.

By repeating this procedure for the non-standard logical constants in $h(S(\underline{p}))$, and by using Theorem 2.3, we arrive at a formula schema $\alpha(\underline{p})$ which only contains the standard constants and which is schematically equivalent to $S(\underline{p})$.[4] Hence we obtain, by repeatedly using Theorem 2.3, the following result: $X \vdash R$ is derivable in the system with arbitrary many logical constants iff $X^\circ \vdash R^\circ$ is derivable for certain X° and R° which are obtained from X and R by successively replacing $S(\underline{T})$ by $\alpha(\underline{T})$ for all non-standard constants in X and R. Since the standard logical constants have no immediate predecessors, Theorem 2.5 implies that $X^\circ \vdash R^\circ$ is derivable by only using primitive rules for the standard logical

[4]More precisely, we need an analogue of Theorem 2.3 which works on the level of formula *schemata* rather than formulae. However, this is immediately obtained from Theorem 2.3 by treating schematic letters as primitive predicates.

constants. Therefore, the reasoning with higher-level rules and arbitrary logical constants can be embedded into intuitionistic logic with the standard constants.[5]

[5]Of course, intuitionistic logic itself can be formulated without using higher-level rules. This result is no argument against considering higher-level rules and uniform introduction and elimination schemata for logical constants. On the contrary, it gives us considerable insight into the expressive strength of intuitionistic logic, which we would not have obtained without the apparatus of higher-level rules.

Chapter 3

Structural Absurdity and Negation

Absurdity $\bot$ was characterized as a logical constant which has no introduction rule, and the "ex falso quodlibet"

$$\bot \Rightarrow p$$

as its elimination schema. It functions as a limiting case, obtained by assuming that $\bot$ has *no* assertibility condition at all, i.e., $\mathbb{D}(\bot)$ is empty. This must be distinguished from the case of the constant $\top$, whose only assertibility condition is the empty set, i.e., for which $\mathbb{D}(\top) = \emptyset$.

There may be doubts of whether the reasoning which justifies $\bot$ as a limiting case is plausible. What one essentially does in this justification is to pass from the idea that logical constants are characterized through certain introduction rules to the idea that the *absence* of such introduction rules is also a way of characterizing them. Correspondingly, the schema of $\bot$-elimination is obtained by omitting from the general schema for elimination rules all premises which refer to assertibility conditions (and thus to premises of introduction rules), so that $\bot$ is the only premise which remains.

From the viewpoint of the "common content"-motivation, one passes from the general idea that for a 0-ary connective S the content of S is the common content of certain sets $\Delta_1, \ldots, \Delta_m$, to $m = 0$ as a limiting case, i.e. to the common content of the elements of an empty set. Formally this may be considered an intersection over an empty set (related to a basic domain of rules), having the set of *all* rules as its result. However, this argument relies on the ex falso quodlibet in the metalanguage: It uses that the condition "for all X (if $X \in \emptyset$ then $X \vdash R$ is derivable)" is fulfilled by *all* rules R. One might argue that in our metalanguage we should not use problematic principles such as ex falso quodlibet, which are for example not accepted in relevance logic.

Therefore, even if one wants to have something like ex falso quodlibet in the object language, one may prefer to simply postulate it without any justification or with a completely different justification, rather than to rely on its justification as a limiting case of an elimination rule without any introduction rule being given. We do not argue here for this position but just describe how it can be spelled out.

The best strategy seems to us to extend the structural framework, and not to destroy the systematics of introduction and elimination rules for logical constants by additional primitive rules. This means that we define a *structural* constant for absurdity and postulate an inference schema, which on the structural level corresponds to ex falso quodlibet. Denoting this structural absurdity by "#", rules are defined as follows:

Each formula is a rule. The structural constant # is a rule. If X is a finite set of rules and A a formula, then $X \Rightarrow A$ and $X \Rightarrow \#$ are rules. (We only consider rules without generality since the problems of generality are independent of the problems discussed here.)

In our formulation of inference schemata for SC1 to SC4 (see Table 1, Ch. 1 § 1), the syntactic variable "A" now stands both for formulae and for #. As a new inference schema we add

$$\frac{X \vdash \#}{X \vdash R}\,(\text{EFQ})\ .$$

When defining logical constants we can introduce a constant F of logical absurdity as corresponding to structural absurdity #, by choosing $\{\#\}$ to be the assertibility condition for F: $\mathbb{D}(\mathsf{F}) = \{\{\#\}\}$. (We use the symbol "$\mathsf{F}$" to distinguish this conceptually new kind of absurdity from $\bot$, for which $\mathbb{D}(\bot) = \emptyset$.) F has the introduction rule

$$\# \Rightarrow \mathsf{F} \qquad (\mathsf{F}\text{-I})$$

and as schema for elimination rules

$$\mathsf{F}, (\# \Rightarrow r) \Rightarrow r, \quad \text{equivalently: } \mathsf{F} \Rightarrow \# \qquad (\mathsf{F}\text{-E})$$

where r is now a schematic variable which may instantiated by formulae or by #.[1] These trivial logical rules for F only serve to connect logical absurdity to structural absurdity for which we have (EFQ) as an inference schema.

In a corresponding way we can characterize minimal and classical logic. *Minimal logic* is obtained by just omitting the structural inference schema (EFQ), i.e., in minimal logic structural absurdity # and logical absurdity F are available with (F-I) and (F-E) as primitive rules, but no specific schema that governs structural absurdity is postulated. *Classical logic* is obtained by adding a schema like

$$\frac{X, (A \Rightarrow \#) \vdash \#}{X \vdash A}\,(\#_c)\ .$$

[1]Structural absurdity # is defined as a constant that may be used in the construction of rules (like the rule arrow) but that is *not* a formula. Therefore the letters $p_1, \ldots, p_n$ in an assertibility condition $\Delta(p_1, \ldots, p_n)$ must not be instantiated by #. When we here allow for the specific letter r to be instantiated by #, we are implicitly using an extended concept of rule schema, in which in addition to schematic letters for formulae (or predicate terms in the general case), schematic letters for other entities like # may also occur. However, since these additional letters are only used in schemata for elimination rules, whereas for assertibility conditions (and thus for schemata for introduction rules) nothing is changed, it would be confusing to give a new definition of rule schema.

The schema

$$\frac{X, (R \Rightarrow \#) \vdash \#}{X \vdash R}\,(\#'_{\mathrm{c}})$$

with R instead of A is not more general than $(\#_{\mathrm{c}})$: Suppose we have derived the sequent $X, (R \Rightarrow \#) \vdash \#$ and R is of the form $Y \Rightarrow B$. Then the sequent $Y, (Y \Rightarrow B), (B \Rightarrow \#) \vdash \#$ can be derived, therefore $Y, (B \Rightarrow \#) \vdash R \Rightarrow \#$ and, together with $X, (R \Rightarrow \#) \vdash \#$, we obtain $X, Y, (B \Rightarrow \#) \vdash \#$. By application of $(\#_{\mathrm{c}})$ we obtain $X, Y \vdash B$ and therefore $X \vdash R$. If one reduces negation $\neg$ to absurdity F by defining $\mathbb{D}(\neg) = \{\{p \Rightarrow \mathsf{F}\}\}$, then one obtains the standard laws of classical propositional logic.

A way of directly defining negation instead of reducing it to F is to enlarge the structural framework in a similar way as we did with $\#$. We define a *structural negation* which intuitively might be thought of as the denial of a formula and denote it by "$-$".[2] Then rules are defined as follows.

Each formula A is a rule. If A is a formula, then $-A$ is a rule. If X is a finite set of rules and A a formula, then $X \Rightarrow A$ and $X \Rightarrow -A$ are rules.

The letter "A" in our inference schemata for SC1-SC4 must now be interpreted as standing both for formulae and for rules of the form $-B$. The following inference schemata governing structural negation are of particular interest:

$$\frac{X, B \vdash C \quad X, B \vdash -C}{X \vdash -B}\,(\mathrm{RA})$$

$$\frac{X \vdash B \quad X \vdash -B}{X \vdash C}\,(\mathrm{ECQ})$$

$$\frac{X, -B \vdash C \quad X, -B \vdash -C}{X \vdash B}\,(\mathrm{RA_c})\ .$$

Here B and C stand for formulae, not for rules of the form $-A$.

Inference schema (RA) leads to minimal logic, (RA) together with (ECQ) to intuitionistic logic and (RA) together with ($\mathrm{RA_c}$) to classical logic. The primitive rules for logical negation are in all cases the same, namely

$$-p \Rightarrow \neg p \qquad (\neg\text{-I})$$

and

$$(\neg p, (-p \Rightarrow r)) \Rightarrow r, \quad \text{equivalently: } \neg p \Rightarrow -p \qquad (\neg\text{-E})$$

[2]This idea goes back to von Kutschera (1969), who uses the sign "$\sim$" as distinguished from logical negation "$\neg$". We shall use "$\sim$" as a second kind of logical negation in relevance logic (see Ch. 5).

where p is a schematic letter for formulae and r is a schematic letter both for formulae A and rules of the form $-A$ (see footnote 1).

Instead of only considering structural negation of formulae, we may also introduce structural negation of rules. The definition of rules would then be the following:

Each formula is a rule. If R is a rule which is not of the form $-R_1$, then $-R$ is a rule. If X is a finite set of rules and R is a rule, then $X \Rightarrow R$ is a rule.

This procedure requires a system like SC2+ or SC3+ (see Ch. 1 § 2), where rules of higher levels may be conclusions of rules. Structural schemata for the structural negation of rules, which correspond to those for the structural negation of formulae, are then

$$\frac{X, R \vdash R' \quad X, R \vdash -R'}{X \vdash {\sim} R}\,(\mathrm{RA}')$$

$$\frac{X \vdash R \quad X \vdash -R}{X \vdash R'}\,(\mathrm{ECQ}')$$

$$\frac{X, -R \vdash R' \quad X, -R \vdash -R'}{X \vdash R}\,(\mathrm{RA_c})\ .$$

If $-R$ is defined as $\{(R \Rightarrow C), (R \Rightarrow -C)\}$ for a fixed formula C, then these inference schemata can be shown to be equivalent to (RA), (ECQ) and ($\mathrm{RA_c}$), respectively. For (ECQ') this is obvious, for (RA') and (RA'_c) this is proved by arguments similar to that for the equivalence of ($\#_\mathrm{c}$) and ($\#'_\mathrm{c}$). Therefore we have not obtained a proper extension of the previous approach.

Something really new is achieved if the structural negation $-R$ of a rule R is not, as before, understood as "R leads to a contradiction", but as "the premises of R can be established whereas the conclusion of R can be refuted". This reading of structural negation was called "direct" by von Kutschera (1969). In von Kutschera (1985) a corresponding "direct logic" is developed in detail, including both proof theory and valuation semantics. Reconstructed in our framework, direct structural negation is characterized by the inference schemata

$$\frac{X \vdash Y \quad X \vdash -R}{\overline{X \vdash -(Y \Rightarrow R)}}$$

(see von Kutschera 1985, p. 35). The rules for logical constants based on such a structural framework do not only contain introduction and elimination rules in our sense but also rules governing the structural negation of logically compound formulae, i.e. refutation rules. Direct logic resembles in some way to systems developed by Fitch (1952) and Nelson (1949) (see Prawitz 1965, Appendix B).

Chapter 4

Logic Programming with Higher-Level Rules

The theory of logic programming is a theory of backward reasoning for inference systems for atomic formulae. In the standard case of definite Horn clause programming, which underlies the programming language PROLOG, only rules of levels 0 and 1 (in our terminology) are considered as primitive. We shall describe a generalization which permits higher-level rules in logic programs and thus extends their means of expression. The basic soundness and completeness theorems for SLD-resolution can also be extended to our general case. To make our presentation self-contained, we state some preliminaries concerning the object language and concerning substitution and unification in § 1, following in most points Lloyd's (1984) textbook. In § 2 basic issues of the theory of programming with definite Horn clauses are developed in purely proof-theoretic terms and not, as usual, by means of Herbrand interpretations. Using a generalization of this method, a theory of higher-level logic programming is outlined in § 3.

§ 1. Preliminaries

In this chapter no bound variables are mentioned. When we speak of *variables*, we always mean *free variables* in the sense of Ch. 1 § 3. We use a language with variables, function symbols and predicate symbols, where 0-place function symbols are individual constants and 0-place predicate symbols are propositional constants.

Terms are variables and expressions of the form $f(t_1, \ldots, t_n)$, where f is an n-place function symbol and $t_1, \ldots, t_n$ are terms (in particular, individual constants are terms).

Atomic formulae, in short *atoms*, are expressions of the form $P(t_1, \ldots, t_n)$, where P is an n-place predicate symbol and $t_1, \ldots, t_n$ are terms (in particular, propositional constants are atoms).

Complex formulae need not be defined, since we shall interpret clauses as rules rather than logically compound expressions. In this and the next section, we use A, B, C for atoms and X, Y, Z for finite sets of atoms. The equality sign "=" is a metalinguistic symbol used to express the identity of sets or of symbols (depending on the context).

Substitutions are treated as formal objects and are metalinguistically denoted by lower case Greek letters. They can be defined as sets of pairs $\{a_1/t_1, \ldots, a_n/t_n\}$ such that the a_i's are all distinct and a_i/t_i for each i $(1 \leq i \leq n)$. A pair a/t in a substitution is called a *binding* for a. If E is an atom, term or rule (rules are considered in the next section), and σ is $\{a_1/t_1, \ldots, a_n/t_n\}$, then $E\sigma$ is $E[a_1, \ldots, a_n/t_1, \ldots, t_n]$, i.e., the result of simultaneously substituting all t_i for the corresponding a_i in E $(1 \leq i \leq n)$. Substitutions in sets of expressions are defined elementwise. The *composition* $\sigma\theta$ of substitutions σ and θ can be defined as an associative operation in such a way that for any expression or set thereof, for which substitution is defined, $E(\sigma\theta) = (E\sigma)\theta$. The union $\sigma \cup \theta$ of substitutions is defined as the usual set union, provided $t_1 = t_2$ if a/t_1 is in σ and a/t_2 is in θ.

If E_1 and E_2 are atoms or terms, then σ is called a *unifier* of E_1 and E_2 if $E_1\sigma = E_2\sigma$. A unifier σ of E_1 and E_2 is called a *most general unifier*, in short: mgu, if for any other unifier θ of E_1 and E_2 there is a substitution δ such that $\theta = \sigma\delta$ (roughly speaking, θ is a specialization of σ).

There is a *unification algorithm* which, given any two atoms or terms E_1 and E_2, constructs an mgu of E_1 and E_2, if E_1 and E_2 are unifiable, and reports failure otherwise. This unification algorithm is the basis of the resolution method developed by Robinson (1965), on which the evaluation of queries in logic programming systems is based.

§ 2. A proof-theoretic view of definite Horn clause programs

In this section we reconstruct the standard theory of definite Horn clause programs in our proof-theoretic context. First we give the following definition:

Each atom is a *definite Horn clause.* If A is an atom and X a finite set of atoms, then $A \longleftarrow X$ is a definite Horn clause. A is called the *head* and X the *body* of $A \longleftarrow X$. A *definite Horn clause program* is a finite set of definite Horn clauses.

Usually, definite Horn clauses are considered logically compound formulae: A clause of the form A is interpreted as the formula $\forall(A)$ and a clause of the form $A \longleftarrow \{B_1, \ldots, B_n\}$ as the formula $\forall((B_1 \,\&\, \ldots \,\&\, B_n) \supset A)$, where $\forall$ followed by a formula denotes the universal closure of this formula. A program then turns out to be a finite set of formulae of first-order logic, and queries in a logic programming system are considered to be questions concerning logical consequences of a program (see Lloyd 1984, Ch. 1).

However, the fragment of first-order logic that is actually needed contains not more than what can be encoded in terms of rules. (This is formally proved in Hallnäs & Schroeder-Heister 1987.) Therefore we propose to consider definite Horn clauses to be rules of levels 0 or 1, writing $X \Rightarrow A$ instead of $A \longleftarrow X$, and a definite Horn clause program to be the set of primitive rules of a formal system. Since we do not deal in this section with higher-level rules nor with assumptions, it would not make much sense to work in a sequent-style framework, since all sequents would be of the form $\emptyset \vdash A$. It is sufficient to consider a calculus whose derivations are trees of atoms and whose only inference schema is that of application of primitive rules:

$$X \Rightarrow A\, \frac{X\sigma}{A\sigma}\,(\mathrm{P}*) \quad (X \Rightarrow A \text{ primitive rule})\ .$$

(This covers

$$A\, \frac{}{A\sigma}$$

as a limiting case, since $\emptyset \Rightarrow A$ is identified with A.) According to (P$*$), variables in a primitive rule are understood as expressing generality. For technical reasons we even allow for $X \Rightarrow A$ to be a *variant* of a primitive rule. A variant is here understood as the result of renaming variables in such a way that different occurrences of a variable a are transformed into different occurrences of a variable a', and occurrences of different variables a and b are transformed into occurrences of different variables a' and b'.[1] This yields no proper generalization of our system but will later on dispense us from explicitly considering renaming substitutions.

We now suppose that a program, viewed as a set of primitive rules, is fixed. The corresponding calculus based on the inference schema (P$*$) is called CP. (The "C" should remind one of "calculus" and the "P" of "PROLOG".) An atom A is called *derivable* in CP, if there is a derivation of A using only (P$*$). A set consisting of derivations of each element of a set of atoms X is called a *derivation of* X. The length of a derivation of X is the sum of the lengths of the derivations of the elements of X. A set of atoms X is called derivable in CP if there is a derivation of X in CP. It is obvious that if A or X is derivable in CP, then so is $A\sigma$ or $X\sigma$, respectively, for any σ.

A set X of atoms is also called a *goal*. A *query* asks *whether* for a certain goal there is a substitution σ such that $X\sigma$ is derivable in CP, and if so, *for which* σ this holds. Each σ, for which this holds, is called a *correct answer substitution* for X (with respect to the program presupposed).[2] A usual representation of queries is

?- X .

A logic programming system should either reply to this query that there is no correct answer substitution, or return a most general set of correct answer substitutions for X. (Here a set of correct answer substitutions is called most general if every correct answer substitution θ can be written as $\sigma\delta$ for some σ in this set.)

The standard method for computing answer substitutions, given a program and a query, is SLD-resolution. We suppose that a *selection function* sel is given, which from any goal

[1]This definition of a variant is different from that in Ch. 1 § 3, where variants were the result of renaming *bound* variables. However, since we do not deal here with bound variables, no confusion can arise. Conceptually, there is no essential difference since variables in program rules are treated as expressing generality.

[2]Unlike Lloyd (1984), we do not assume that the bindings in correct answer substitutions for X are restricted to variables actually occurring in X.

X selects an element sel(X). An *SLD-derivation* with respect to a selection function sel (and with respect to the program presupposed) is a sequence of goals which is generated by the inference schema

$$X \Rightarrow A\,\frac{Y \dot{\cup} \{B\}}{Y\sigma \cup X\sigma}\,\sigma \quad (\mathrm{P}*)' \;,$$

where $X \Rightarrow A$ is a variant of a primitive rule, $B = \mathrm{sel}(Y \dot{\cup} \{B\})$ and σ is an mgu of B and A. (When we write "$X \dot{\cup} Y$" rather than "$X \cup Y$", this is to express that X and Y are assumed to be disjoint.) Here σ is called the mgu *used* at the application of $(\mathrm{P}*)'$. The step $(\mathrm{P}*)'$ can be rephrased as follows: Given a goal Z, we select an element B of Z according to the selection function sel, where $Y = Z \setminus \{B\}$. Then we try to unify B with the head A of a rule $X \Rightarrow A$ which is either primitive or a variant of a primitive rule, replace the selected element B in Z by the premises X of this rule, and substitute the result by σ. Given an order in which to try primitive rules, this can be performed by a machine by primarily using the unification algorithm. (The problems of search strategies connected with the order of primitive rules cannot be discussed here; see Lloyd 1984, 10.)

An SLD-derivation *for* X is an SLD-derivation which starts with X, and a *successful* SLD-derivation for X is an SLD-derivation for X which ends with the empty set. If $\sigma_1, \ldots, \sigma_n$ is the sequence of mgu's of a successful SLD-derivation for X, σ_i being used in the i-th step from the top ($1 \leq i \leq n$), then the composition $\sigma_1\sigma_2 \ldots \sigma_n$ is called the *computed answer substitution* of this SLD-derivation for X.

Now we can prove the soundness and completeness of the theory of definite Horn clause programs in a straightforward way, by only using proof-theoretic methods.

Theorem 4.1 (Soundness of SLD-resolution) If a successful SLD-derivation for Z with respect to an arbitrary selection function is given with computed answer substitution τ, then $Z\tau$ is derivable in CP. (I.e., each computed answer substitution is correct.)

Proof Induction on the length of successful SLD-derivations. If their length is 1, they just consist of the empty set, and nothing is to be proved. If they are of length $n + 1$, starting with a step

$$X \Rightarrow A\,\frac{Y \dot{\cup} \{B\}}{Y\sigma \cup X\sigma}\,\sigma \quad (\mathrm{P}*)' \;,$$

then by omitting this first step, we obtain a successful SLD-derivation of $Y\sigma \cup X\sigma$ of length n with computed answer substitution δ. By induction hypothesis, we obtain a derivation of $Y\sigma\delta \cup X\sigma\delta$ in CP. Therefore, by using the inference schema $(\mathrm{P}*)$ (applying $X \Rightarrow A$ to $X\sigma\delta$), we obtain a derivation of $Y\sigma\delta \cup \{A\sigma\delta\}$ in CP which is the same as $Y\sigma\delta \cup \{B\sigma\delta\}$, since σ is a unifier of A and B. Now $\sigma\delta$ is the computed answer substitution of the successful SLD-derivation of length $n + 1$. □

We say that σ and θ *agree* on Z, if σ and θ become identical after deleting all bindings for variables not in Z.

Theorem 4.2 (Strong completeness of SLD-resolution) Let an arbitrary selection function sel be given. If $Z\tau$ is derivable in CP, then there is a substitution δ and a successful SLD-derivation for Z with respect to sel with computed answer substitution σ such that τ and $\sigma\delta$ agree on Z. (I.e., each correct answer substitution for X is a specialization of a computed answer substitution for X, when restricted to the variables occurring in X.)

Proof by induction on the length of derivations of sets $Z\tau$ in CP. If $Z\tau$ and therefore Z is empty, then we can take the empty set as a successful SLD-derivation and δ to be τ. Now suppose that Z is nonempty, $B = \mathrm{sel}(Z)$ and $Y = Z \setminus \{B\}$, i.e., the derivation of $Z\tau$ in CP consists of derivations of $Y\tau$ and $B\tau$. We consider the derivation of $B\tau$. Suppose this derivation proceeds in the last step according to schema (P$*$), i.e.

$$X \Rightarrow A\,\frac{X\theta}{A\theta}$$

where $A\theta = B\tau$. By renaming variables in $X \Rightarrow A$ we can replace this step by

$$X' \Rightarrow A'\,\frac{X'\theta'}{A'\theta'}$$

where $X'\theta' = X\theta$, $A'\theta' = A\theta = B\tau$, and $X' \Rightarrow A'$ is a variant of $X \Rightarrow A$ such that $X' \Rightarrow A'$ and Z have no variables in common, θ' contains no bindings for variables in Z and τ no bindings for variables in $X' \Rightarrow A'$. Thus $Z\tau = Z(\tau \cup \theta')$ and $(X' \Rightarrow A')\theta' = (X' \Rightarrow A')(\tau \cup \theta')$. In particular, $\tau \cup \theta'$ is a unifier of B and A'. Let μ be an mgu of B and A'. Then $\tau \cup \theta' = \mu\tau_1$ for some τ_1. Thus we can write the inference step under consideration as

$$X' \Rightarrow A'\,\frac{X'\mu\tau_1}{B\mu\tau_1}\ .$$

By omitting this final step in the derivation of $B\tau$ and leaving the derivation of $Y\tau$ unchanged, we obtain a derivation of $Y\mu\tau_1 \cup X'\mu\tau_1$, which is by 1 shorter than that of $Z\tau(= Z\mu\tau_1)$. We may assume by induction hypothesis (with respect to the derivation of $(Y\mu \cup X'\mu)\tau_1$) that we are already given a substitution δ and an SLD-derivation of $Y\mu \cup X'\mu$ with computed answer substitution σ_1 such that τ_1 and $\sigma_1\delta$ agree on $Y\mu \cup X'\mu$. By adding the step

$$X' \Rightarrow A'\,\frac{Y \,\dot\cup\, \{B\}}{Y\mu \cup X'\mu}\,\mu \quad (\mathrm{P}*)'$$

at the top of this SLD-derivation, we obtain an SLD-derivation for Z with computed answer substitution $\mu\sigma_1$. Since τ and $\mu\tau_1$ agree on Z, $\mu\sigma_1\delta$ and τ agree on Z, too. □

This theorem is called "strong" completeness theorem, because it is independent of the choice the selection function.

§ 3. Higher-level rules in programs

The extension of definite Horn clause logic described in the following uses higher-level rules over atoms. We do not consider generality in rules here, so that our approach corresponds to the propositional case of Ch. 1 § 1. We will base it on SC4 as the most appropriate system for backward reasoning. However, primitive rules are implicitly understood as generalized, so that we shall use (App-P$*$) rather than (App-P) as the schema for the application of primitive rules. Therefore, as in the previous section, our propositional framework is not "pure".

A full treatment of logic programming with higher-level rules including generality requires an extension of the algorithmic aspects, especially of the unification algorithm. It would therefore exceed the scope of this chapter, whose purpose is mainly to illustrate the application of a structural framework with higher-level rules to theories of reasoning with atomic formulae. The general theory, which also contains an extension of logic programming in which clauses for predicates are given a definitional reading, is being developed by Hallnäs and Schroeder-Heister (1987, 1988).

The sort of rules we are dealing with here is defined as follows:

Each atom is a rule. If X is a nonempty finite set of rules and A an atom, then $X \Rightarrow A$ is a rule. Variants of rules are defined as in § 2.

All notational conventions of Ch. 1 are valid for this section. In particular, R stands for rules and X, Y, and Z stand for finite sets of rules (and no longer only for sets of atoms as in § 2). A sequent is of the form $X \vdash R$. Finite sets of sequents are denoted by Γ and Σ.

A *generalized Horn clause program*, in short: program, is a finite set of rules. We consider a fixed program to be given, whose rules are understood as the primitive rules of a formal system. (They are also called "program rules" as opposed to "assumption rules".) This formal system is a sequent calculus corresponding to SC4, called SCP, and has the following inference schemata:

$$\frac{}{X, A \vdash A}\,(\mathrm{Ini}) \qquad Y \Rightarrow A\,\frac{X \vdash Y\sigma}{X \vdash A\sigma}\,(\text{App-P}*)$$

$$\frac{X \vdash Y \quad X, A \vdash R}{X, (Y \Rightarrow A) \vdash R}\,(\Rightarrow\vdash) \qquad \frac{X, Y \vdash A}{X \vdash Y \Rightarrow A}\,(\Rightarrow +)\ ,$$

where $Y \Rightarrow A$ is a primitive rule or a variant thereof. Again, permitting the use of variants of primitive rules in applications of (App-P$*$) yields no proper extension, but is just a convenient way of avoiding explicit mentioning of renaming substitutions.

Since there are no eigenvariable conditions in the inference schemata, it is trivial that if $X \vdash R$ is derivable in SCP, then so is $X\sigma \vdash R\sigma$ for any σ. It is one of the great advantages of working with sequents that we immediately obtain this substitution property. In that way, we have a natural distinction between program rules which are understood as generalized, and assumptions rules which are instantiated when the sequent, in whose antecedent they occur, is instantiated, i.e., between the derivability of a sequent $X, R \vdash R'$ with respect to a certain program Pr and the derivability of the sequent $X \vdash R'$ with respect to the enlarged program $\mathrm{Pr} \cup \{R\}$.[3]

A *goal* is now a finite set Γ of sequents. Similar to the conventions of the previous section, a derivation of Γ in SCP is defined as a set consisting of derivations of each element of Γ in SCP. If Γ is $\{X_1 \vdash R_1, \dots, X_n \vdash R_n\}$, then $\Gamma\sigma$ is defined as $\{X_1\sigma \vdash R_1\sigma, \dots, X_n\sigma \vdash R_n\sigma\}$ and $\emptyset\sigma$ as $\emptyset$. A *query* asks whether for some goal Γ there is a σ such that $\Gamma\sigma$ is derivable in SCP, and if there is such a σ, for which σ this holds. It may again be denoted by

$$?\text{-}\ \Gamma\ .$$

The evaluation procedure for queries is described by defining an extended notion of SLD-derivation. SLD-derivations are sequences of goals (now understood as sets of sequents) with respect to a selection function sel which selects from each goal a sequent. An SLD-derivation of Γ is an SLD-derivation starting with Γ, a successful SLD-derivation of Γ is an SLD-derivation of Γ which ends with the empty set. The inference steps according to which SLD-derivations are constructed, are given by the following four inference schemata, which correspond to the inference schemata of SCP. The goals above the inference line are all of the form $\Sigma \dot\cup \{Z \vdash R\}$, and it is always assumed that $Z \vdash R$ is the selected element of this goal, i.e., $\mathrm{sel}(\Sigma \dot\cup \{Z \vdash R\}) = Z \vdash R$.

$$\frac{\Sigma \dot\cup \{X, A \vdash B\}}{\Sigma\sigma}\,\sigma \quad (\mathrm{Ini})' \qquad\qquad Y \Rightarrow A\, \frac{\Sigma \dot\cup \{X \vdash B\}}{\Sigma\sigma \cup (X\sigma \vdash Y\sigma)}\,\sigma \quad (\mathrm{App\text{-}P}{*})'$$

where σ is an mgu of A and B and $Y \Rightarrow A$ is a primitive rule or a variant thereof

[3]This advantage becomes obvious when we compare our approach with that of Gabbay and Reyle (1984), in which definite Horn clause programs are enlarged with logical implication. Though Gabbay and Reyle work in a clauses-as-formulae framework whereas we adhere to the clauses-as-rules idea, there is a certain relationship between the two theories, since higher-level rules can be translated into conjunction-implication formulae (see Ch. 2 § 3). However, Gabbay and Reyle do not use a sequent-style framework but instead two different sorts of variables in program rules with different substitutivity conditions. These substitutivity conditions are formulated in such a way that our distinction between the derivability of $\emptyset \vdash R$ with respect to $\mathrm{Pr} \cup X$ and the derivability of $X \vdash R$ with respect to Pr has a counterpart in a distinction by Gabbay and Reyle between the derivability of R with respect to $\mathrm{Pr} \cup X$ and the derivability of R with respect to $\mathrm{Pr} \cup X^*$ for some X^*, where X^* and X differ in the sort of variables they contain. This way of proceeding lacks a sufficient intuitive foundation as compared to the clear notion of the derivability of a sequent. Furthermore, it leads to very complicated proofs of soundness and completeness of the query evaluation procedure. And finally, Gabbay and Reyle's system is difficult to implement, so that they consider in addition a weaker version of their system, in which variables of the different sorts must not occur in one and the same clause. Our system, which is based on a simple calculus of sequents, can be implemented relatively easily (if one disregards the problems of search strategies). An implementation in LISP has been written for experimental purposes by Aronsson and Montelius (1986). Quite surprisingly, Gabbay and Reyle (1984) speak of "quantified" PROLOG when they mean just logic programming with free variables in rules. We prefer to reserve such a term for a version in which the premises of rules may contain generality operators. In Miller's (1986) extension of definite Horn clause programming, in which queries may be made under assumptions, this problem is avoided by not allowing that these assumptions contain variables.

$$\frac{\Sigma \dot\cup \{X,(Y \Rightarrow A) \vdash R\}}{\Sigma \cup (X \vdash Y) \cup \{X, A \vdash R\}} (\Rightarrow\vdash)' \qquad\qquad \frac{\Sigma \dot\cup \{X \vdash Y \Rightarrow A\}}{\Sigma \cup \{X, Y \vdash A\}} (\Rightarrow +)' \ .$$

At applications of (Ini)$'$ and (App-P$*$)$'$, the mgu σ of A and B which is mentioned on the right of the inference line, is called the mgu *used* at this application. At applications of $(\Rightarrow\vdash)$ and $(\Rightarrow +)$ in an SLD-derivations no mgu is used. If $\sigma_1, \ldots, \sigma_n$ is the sequence of mgu's used in a successful SLD-derivation for Γ, counted from the top to the bottom, then $\sigma_1 \ldots \sigma_n$ is the computed answer substitution of this SLD-derivation for X (with respect to the selection function sel).

It is obvious that concerning search strategies there are now many more options. Not only the order is relevant in which unification with conclusions of primitive rules is tried according to (App-P$*$)$'$, but also whether and in which order attempts at unifying succedents of sequents with atoms in the antecedent are made according to (Ini)$'$; furthermore, whether and in which order rules in antecedents or succedents of sequents are evaluated according to $(\Rightarrow\vdash)'$ and $(\Rightarrow +)'$.

The soundness and completeness theorems for this generalized form of SLD-resolution with respect to SCR follow the pattern of the proofs of Theorems 4.1 and 4.2. However, more case distinctions are necessary, since more inference schemata are to be considered.

Theorem 4.3 (Soundness of generalized SLD-resolution) If a successful SLD-derivation for Γ with respect to an arbitrary selection function is given with computed answer substitution τ, then $\Gamma\tau$ is derivable in SCP. (I.e., each computed answer substitution is correct.)

Proof Induction on the length of successful SLD-derivations. If such a derivation is of length 1, it just consist of the empty set, and nothing is to be proved. If it is of length $n+1$, starting with a step (Ini)$'$:

$$\frac{\sigma \dot\cup \{X, A \vdash B\}}{\Sigma\sigma} \sigma \quad (\text{Ini})' \ ,$$

then by omitting this first step, we obtain a successful SLD-derivation of $\Sigma\sigma$ with computed answer substitution δ. By induction hypothesis, we obtain a derivation of $\Sigma\sigma\delta$ in SCP. By using (Ini), we obtain a derivation of $\Sigma\sigma\delta \cup \{X\sigma\delta, A\sigma\delta \vdash A\sigma\delta\}$, which is the same as $\Sigma\sigma\delta \cup \{X\sigma\delta, A\sigma\delta \vdash B\sigma\delta\}$, since σ is a unifier of A and B. Now $\sigma\delta$ is the computed answer substitution of the successful SLD-derivation of length $n+1$.

If the successful SLD-derivation of length $n+1$ starts with a step (App-P$*$)$'$:

$$Y \Rightarrow A \, \frac{\Sigma \dot\cup \{X \vdash B\}}{\Sigma\sigma \cup (X\sigma \vdash Y\sigma)} \sigma \quad (\text{App-P}*)' \ ,$$

then by omitting this first step, we obtain a successful SLD-derivation of $\Sigma\sigma \cup (X\sigma \vdash Y\sigma)$ with computed answer substitution δ. By induction hypothesis, we obtain a derivation of $\Sigma\sigma\delta \cup (X\sigma\delta \vdash Y\sigma\delta)$ in SCP. By using (App-P$*$), we obtain a derivation of $\Sigma\sigma\delta \cup \{X\sigma\delta \vdash A\sigma\delta\}$, which is the same as $\Sigma\sigma\delta \cup \{X\sigma\delta \vdash B\sigma\delta\}$, since σ is a unifier of A and B.

Now $\sigma\delta$ is the computed answer substitution of the successful SLD-derivation of length $n+1$.

If the successful SLD-derivation of length $n+1$ starts with one of the steps $(\Rightarrow\vdash)'$ or $(\Rightarrow +)'$:

$$\frac{\Sigma \,\dot\cup\, \{X, (Y \Rightarrow A) \vdash R\}}{\Sigma \cup (X \vdash Y) \cup \{X, A \vdash R\}}\,(\Rightarrow\vdash)' \qquad\qquad \frac{\Sigma \,\dot\cup\, \{X \vdash Y \Rightarrow A\}}{\Sigma \cup \{X, Y \vdash A\}}\,(\Rightarrow +)'\ ,$$

let in each case the goal above the inference line be called Σ_1 and that below the inference line be called Σ_2. Then by omitting the first step of the SLD-derivation, we obtain a successful SLD-derivation of Σ_2 with computed answer substitution δ. This is also the computed answer substitution of the successful SLD-derivation of Σ_1, since no mgu is used at applications of $(\Rightarrow\vdash)'$ or $(\Rightarrow +)'$. By induction hypothesis, we obtain a derivation of $\Sigma_2\delta$ in SCP, and by using $(\Rightarrow\vdash)$ or $(\Rightarrow +)$, respectively, we obtain a derivation of $\Sigma_1\delta$ in SCP. □

Substitutions σ and θ are said to agree on a goal Γ, if σ and θ become identical after deleting all bindings for variables not in Γ.

Theorem 4.4 (Strong completeness of generalized SLD-resolution) Let an arbitrary selection function sel be given. If $\Gamma\tau$ is derivable in SCP, then there is a substitution δ and a successful SLD-derivation for Γ with respect to sel with computed answer substitution δ such that τ and $\sigma\delta$ agree on Γ.

Proof by induction on the length of derivations of goals $\Gamma\tau$ in SCP. If $\Gamma\tau$ and therefore Γ is empty, then we can take the empty set as a successful SLD-derivation and δ to be τ. Now suppose that Γ is nonempty and $\Sigma = \Gamma \setminus \mathrm{sel}(\Gamma)$, i.e., the derivation of $\Gamma\tau$ in SCP consists of derivations of $\Sigma\tau$ and $(\mathrm{sel}(\Gamma))\tau$. We distinguish cases according to the inference schema applied in the last step of the derivation of $(\mathrm{sel}(\Gamma))\tau$.
(Ini) The selected sequent is $X, A \vdash B$ and the last step has the form

$$\frac{}{X\tau, A\tau \vdash B\tau}\,(\mathrm{Ini})$$

where τ is a unifier of A and B. Let μ be an mgu of A and B such that $\tau = \mu\tau_1$. Then we have a derivation of $\Sigma\mu\tau_1$ in SCP, which is by 1 shorter than that of $\Sigma\mu\tau_1 \cup \{X\mu\tau_1, A\mu\tau_1 \vdash B\mu\tau_1\}$ $(= \Gamma\tau)$. Thus by induction hypothesis we obtain a substitution δ and a successful SLD-derivation of $\Sigma\mu$ with computed answer substitution σ_1 such that τ_1 and $\sigma_1\delta$ agree on $\Sigma\mu$. Therefore by adding the step

$$\frac{\Sigma \,\dot\cup\, \{X, A \vdash B\}}{\Sigma\mu}\,\mu \quad (\mathrm{Ini})'$$

at the top of this SLD-derivation, we obtain an SLD-derivation for Γ with computed answer substitution $\mu\sigma_1$. Since τ and $\mu\tau_1$ agree on Γ, $\mu\sigma_1\delta$ and τ agree on Γ, too.
(App-P$*$) The selected sequent is $Y \vdash B$ and the last step has the form

$$X \Rightarrow A\, \frac{Y\tau \vdash X\theta}{Y\tau \vdash A\theta}\,(\text{App-P}*)$$

where $A\theta = B\tau$. By renaming variables in $X \Rightarrow A$ we can replace this step by

$$X' \Rightarrow A' \, \frac{Y\tau \vdash X'\theta'}{Y\tau \vdash A'\theta'} \text{ (App-P}*)$$

where $X'\theta' = X\theta, A'\theta' = A\theta = B\tau$, and $X' \Rightarrow A'$ is a variant of $X \Rightarrow A$ such that $X' \Rightarrow A'$ and Γ have no variables in common, θ' contains no bindings for variables in Γ and τ no bindings for variables in $X' \Rightarrow A'$. Thus $\Gamma\tau = \Gamma(\tau \cup \theta')$ and $(X' \Rightarrow A')\theta' = (X' \Rightarrow A')(\tau \cup \theta')$. In particular, $\tau \cup \theta'$ is a unifier of B and A'. Let μ be an mgu of B and A'. Then $\tau \cup \theta' = \mu\tau_1$ for some τ_1. Thus we can write the inference step under consideration as

$$X' \Rightarrow A' \, \frac{Y\mu\tau_1 \vdash X'\mu\tau_1}{Y\mu\tau_1 \vdash B\mu\tau_1} \; .$$

By omitting this final step in the derivation of $Y\tau \vdash B\tau$ and leaving the derivation of $\Sigma\tau$ unchanged, we obtain a derivation of $\Sigma\mu\tau_1 \cup (Y\mu\tau_1 \vdash X'\mu\tau_1)$, which is by 1 shorter than that of $\Gamma\tau$ ($= \Gamma\mu\tau_1$). We may assume by induction hypothesis (with respect to the derivation of $(\Sigma\mu \cup (Y\mu \vdash X'\mu))\tau_1$) that we are already given a substitution δ and an SLD-derivation of $\Sigma\mu \cup (Y\mu \vdash X'\mu)$ with computed answer substitution σ_1 such that τ_1 and $\sigma_1\delta$ agree on $\Sigma\mu \cup (Y\mu \vdash X'\mu)$. By adding the step

$$X' \Rightarrow A' \, \frac{\Sigma \,\dot\cup\, \{Y \vdash B\}}{\Sigma\mu \cup (Y\mu \vdash X'\mu)} \, \mu \quad \text{(App-P}*)'$$

at the top of this SLD-derivation, we obtain an SLD-derivation for Γ with computed answer substitution $\mu\sigma_1$. Since τ and $\mu\tau_1$ agree on Γ, $\mu\sigma_1\delta$ and τ agree on Γ, too.

($\Rightarrow \vdash$) The selected sequent is $X, (Y \Rightarrow A) \vdash R$ and the last step has the form

$$\frac{X\tau \vdash Y\tau \qquad X\tau, A\tau \vdash R\tau}{X\tau, (Y\tau \Rightarrow A\tau) \vdash R\tau} \, (\Rightarrow \vdash) \; .$$

By induction hypothesis we may assume that a successful SLD-derivation of $\Sigma \cup (X \vdash Y) \cup \{X, A \vdash R\}$ is given with computed answer substitution σ such that τ and $\sigma\delta$ agree on $\Sigma \cup (X \vdash Y) \cup \{X, A \vdash R\}$ for some δ. By adding the step $(\Rightarrow \vdash)$ at the top of this SLD-derivation, we obtain an SLD-derivation of Γ with computed answer substitution σ such that τ and $\sigma\delta$ agree on Γ.

($\Rightarrow +$) Analogously. □

Chapter 5

A Structural Framework for Relevance Logic

In this chapter we only deal with rules without generality. This is because the central problems of relevance logic arise at the propositional level. In § 1 we present an enlarged framework for relevance logic, introducing a new sort of structure for higher-level rules which extends Read's and Slaney's[1] notion of a "bunch" of formulae to that of a bunch of extensional and intensional rules. In § 2 logical constants of positive relevance logic are treated in a uniform way, and in § 3 it is discussed how negation and absurdity can be added to this framework. The resulting system, when restricted to the standard constants of relevance logic, is shown in § 4 to be equivalent to a calculus for relevance logic by Read and Slaney. In § 5 we sketch how S4-modality can be added to the framework. The structural framework we propose uses rules whose conclusions may themselves be arbitrary rules. We present it as an extension of our formalism SC3+ (see Ch. 1 § 2 and Table 1), which is best suited for our purpose.

§ 1. Bunches of higher-level rules

In our structural framework with higher-level rules a variety of logical constants could be characterized in a uniform way. Of course, this did not comprise all possible logical constants. On the contrary, it could be shown that every constant which can be characterized in this framework can be defined in terms of the intuitionistic standard connectives $\&$, $\vee$, $\supset$ and $\bot$. If we want to characterize further logical constants, we must extend our framework in some way. The extension we describe in the following allows us to treat constants of relevance logic in nearly the same manner as "ordinary" constants were treated in Ch. 2.

[1]In the following, when referring to Read and Slaney without further specification, we mean the works Slaney (1984, 1986), Read (1984, 1988 Ch. 4), Slaney and Martin (1988). The term "bunch" was first used by Slaney.

One basic idea leading to relevance logic can be stated as follows: Sometimes we do not only want to know whether A follows from *some subset* of X, but also whether A follows from X in the sense that *all* elements of X are relevant for A.[2] This idea could easily be incorporated in our structural framework by simply omitting the inference schema of thinning from SC3+. However, this procedure would be not an extension or *refinement* of our structural framework, but a *restriction*. We would not have more than before, but we would obtain something different. The comma and the rule arrow would acquire a different meaning. Correspondingly, the logical constants & and $\supset$, which on the logical side express the contents of the comma and the rule arrow, would mean something different. It is of course not undesirable that we are able to define a new sort of implication (called relevant implication and denoted by "$\rightarrow$"), and a new sort of conjunction (called "fusion" and denoted by "$\circ$"). But we want these constants not in the place of ordinary conjunction and implication, but *in addition* to them. With respect to conjunction and fusion this is a common opinion in the literature on relevance logics—one always wants to have "extensional" conjunction & in the system (see e.g. Anderson and Belnap 1975, Routley et al. 1982). With respect to extensional versus relevant implication there is no such common opinion, since relevant implication is often considered the "right" implication that must replace ordinary implication. We do not speculate here about which implication is "right" in some sense but simply consider it a reasonable goal to have a framework in which both relevant and non-relevant operators can be defined, and which in respect to the non-relevant part simply contains what we had before.

We shall rely on an idea, which goes back to Dunn (1975) (see also Giambrone 1983) and was developed by Read and Slaney (see footnote 1) for a sequent-style system which, as based on introductions and eliminations of logical constants rather than introductions to the right and left of the turnstile, is near to natural deduction. The idea is simply to extend the means for combining premises or assumptions by using the semicolon ";" in addition to the comma ",", where the comma stands for non-relevant and the semicolon for relevant combination. The most elegant way of handling these combinations is to treat them as binary operations, building up objects which, following Read and Slaney, will be called "bunches". In particular, in premises of a rule or in antecedents of a sequent the comma is no longer considered to divide the elements of a finite set.

The central extension we undertake as compared to the Read/Slaney approach is that we do not only combine formulae und use bunches of formulae as antecedents of sequents, but also define rules of higher levels by use of bunches as premises, so that rules may occur within bunches. Corresponding to the two combinations denoted by the comma and the semicolon, we use a double rule arrow "$\Rightarrow$" (as before) and a triple rule arrow "$\Rrightarrow$". When we consider logical constants, the double arrow will be related to ordinary implication (as before) and the triple arrow to relevant implication.

Since bunches are not sets, the empty set is no candidate for an empty bunch. Instead, we need two objects as empty bunches, one corresponding to the semicolon (denoted by "$\curlyvee$") and one corresponding to the comma (denoted by "$\emptyset$"). Note that in this chapter

[2]The second basic idea of relevance logic, which is the rejection of the "ex falso quodlibet" will be treated in § 3 below.

$\emptyset$ is *not* used as the sign for the empty set but as a specific object (although it functions similarly to the empty set).

Our definition of rules and bunches runs as follows:

(i) Every formula is a rule of level 0.

(ii) $\emptyset$ and $\curlyvee$ are bunches of level 0.

(iii) Every rule of level n is a bunch of level n.

(iv) If X and Y are bunches of levels n_1 and n_2, respectively, then (X, Y) and $(X; Y)$ are bunches of level $\max(n_1, n_2)$.

(v) If X is a bunch of level n_1 and R a rule of level n_2, then $(X \Rightarrow R)$ and $(X \Rrightarrow R)$ are rules of level $\max(n_1, n_2) + 1$.

The metalinguistic notations and conventions are as before, except that "U", "V", "X", "Y", "Z" now stand for bunches rather than sets of rules.

A rule of the form $X \Rrightarrow R$ is called *intensional*; one of the form $X \Rightarrow R$ *extensional*. Similarly, a bunch which is an intensional rule or is of the form $\curlyvee$ or $(X; Y)$ is called intensional; one which is an extensional rule or is of the form $\emptyset$ or (X, Y) is called extensional. A rule or bunch which is just a formula A is called both extensional and intensional.[3] We do not formally identify $\emptyset \Rightarrow R$ and $\curlyvee \Rrightarrow R$ with R, not even if R is a formula. (Lemma 5.2 will show that all these rules are equivalent.)[4]

Sequents are of the form $X \vdash R$ where X is a bunch and R a rule. The intuitive meaning of (X, Y) in the antecedent of a sequent is that X *or* Y *or* both are relevant for the succedent, whereas $(X; Y)$ expresses that both X and Y are relevant. Which parts of X and Y are relevant if X or Y is relevant, depends on the internal structures of X and Y. Of course, this motivation is rather vague. The precise meaning is given to those combinations by certain inference schemata. Roughly speaking, we will allow for thinning in connection with the comma, but not in connection with the semicolon.

As said at the beginning, we present our sequent-style system for relevance logic, which is called SCR, in a way that corresponds to SC3+. First we state those schemata which govern the combination of bunches in antecedents of sequents and have no direct analogue in SC3+ (except thinning).

$$\frac{X, (Y, Z) \vdash R}{\overline{(X, Y), Z \vdash R}}\,(\mathrm{Assoc_e}) \qquad\qquad \frac{X; (Y; Z) \vdash R}{\overline{(X; Y); Z \vdash R}}\,(\mathrm{Assoc_i})$$

[3]It should be clear that in our proof-theoretic framework "extensional" does not mean anything that has to do with truth-functionality.

[4]In Ch. 1 § 1 we identified A with $\emptyset \Rightarrow A$ to obtain a more elegant formulation of SC1, SC1′ and SC4. Since we here work with a system related to SC3+, there is no need for such an identification.

$$\begin{array}{c} X, Y \vdash R \\ \hline\hline Y, X \vdash R \end{array}\,(\mathrm{Comm_e}) \qquad \begin{array}{c} X; Y \vdash R \\ \hline\hline Y; X \vdash R \end{array}\,(\mathrm{Comm_i})$$

$$\begin{array}{c} X, X \vdash R \\ \hline\hline X \vdash R \end{array}\,(\mathrm{Contr_e}) \qquad \begin{array}{c} X; X \vdash R \\ \hline\hline X \vdash R \end{array}\,(\mathrm{Contr_i})$$

$$\begin{array}{c} \emptyset, X \vdash R \\ \hline\hline X \vdash R \end{array}\,(\mathrm{Id_e}) \qquad \begin{array}{c} \Upsilon; X \vdash R \\ \hline\hline X \vdash R \end{array}\,(\mathrm{Id_i})$$

$$\frac{X \vdash R}{X, Y \vdash R}\,(\mathrm{Thin_e})$$

These schemata state that for both the combinations with the comma and with the semicolon we have associativity, commutativity and contraction as we have for sets. Thinning is allowed only with respect to the comma, not with respect to the semicolon. In view of the associativity schemata we can use notations like X, Y, Z or $X; Y; Z$ for bunches which are built up by iterated use of only the comma or the semicolon.

The inference schemata governing the semicolon do not represent the only possible or reasonable choice. We have chosen inference schemata according to which the comma and the semicolon only differ with respect to whether thinning is allowed or not. When enlarged with rules for logical constants, our system leads to the positive fragment of the relevance logic **R** (see Anderson and Belnap 1975). Other structural inference schemata would lead to other systems of relevance logic (see Read and Slaney). Since the treatment of these different systems is not directly connected to our approach of rules of higher levels, we do not discuss them here.

We postulate as inference schemata governing the rule arrows:

$$\begin{array}{c} X, Y \vdash R \\ \hline\hline X \vdash Y \Rightarrow R \end{array}\,(\Rightarrow) \qquad \begin{array}{c} X; Y \vdash R \\ \hline\hline X \vdash Y \Rrightarrow R \end{array}\,(\Rrightarrow)\ .$$

In other words, for the double arrow we have the same schemata as in SC3+, for the triple arrow we have a schema which is associated with the semicolon. The idea behind $(\Rrightarrow)$ is the following: To establish an extensional rule $Y \Rightarrow R$ under assumptions X it is necessary and sufficient to derive its conclusion from X or Y or both. To establish an intensional rule $Y \Rrightarrow R$ under assumptions, it is necessary and sufficient to derive its conclusion from its premises, where both the premises and the assumptions are relevant for the conclusion. According to our conventions, $(\Rightarrow +)$ denotes the direction of $(\Rightarrow)$ from above to below and $(\Rightarrow -)$ the converse one, and similarly for $(\Rrightarrow)$.

Since the arrows "$\Rightarrow$" and "$\Rrightarrow$" may occur mixed within rules, it is not possible in general to work only with formulae as conclusions of rules. For example, a rule of the form $Y \Rrightarrow (Z \Rightarrow A)$ cannot always be replaced by an equivalent rule of the form $U \Rightarrow A$ or $U \Rrightarrow A$ for some U. Only rules in which only "$\Rightarrow$" or only "$\Rrightarrow$" occur can be reduced to rules with formulae as conclusions. Therefore the results of Ch. 1 § 2 do not pertain to the present case.[5] The inference schemata

$$\frac{}{R \vdash R}\,(\mathrm{Refl}) \qquad\qquad \frac{X \vdash R \quad R \vdash R'}{X \vdash R'}\,(\mathrm{Trans})$$

are chosen as basic inference schemata as in SC3+. The handling of primitive rules will be described below. On the basis of the inference schemata defined so far, we can already prove the following results.

Lemma 5.1 The following inference schemata are admissible in every extension of the system developed so far by additional inference schemata.

$$\text{(i)}\ \frac{\emptyset \vdash R}{\Upsilon \vdash R} \qquad \text{(ii)}\ \frac{X;Y \vdash R}{X,Y \vdash R} \qquad \text{(iii)}\ \begin{array}{c} \emptyset \vdash X \Rightarrow R \\ \hline\hline \Upsilon \vdash X \Rrightarrow R \end{array}$$

$$\text{(iv)}\ \begin{array}{c} \emptyset \vdash X \Rrightarrow R \\ \hline\hline \Upsilon \vdash (\emptyset; X) \Rrightarrow R \end{array} \qquad \text{(v)}\ \begin{array}{c} \Upsilon \vdash X \Rightarrow R \\ \hline\hline \emptyset \vdash (\Upsilon, X) \Rightarrow R \end{array}$$

$$\text{(vi)}\ \begin{array}{c} \emptyset \vdash X \Rrightarrow R \\ \hline\hline \emptyset \vdash (\emptyset; X) \Rightarrow R \end{array} \qquad \text{(vii)}\ \begin{array}{c} \Upsilon \vdash X \Rightarrow R \\ \hline\hline \Upsilon \vdash (\Upsilon, X) \Rrightarrow R \end{array}$$

Proof Consider the following derivations:

$$\text{(i)}\ \cfrac{\cfrac{\emptyset \vdash R}{\emptyset, \Upsilon \vdash R}\,(\mathrm{Thin_e})}{\Upsilon \vdash R}\,(\mathrm{Comm_i})\text{ and }(\mathrm{Id_i})$$

$$\text{(ii)}\ \begin{array}{cl} X;Y \vdash R & \\ \hline X \vdash Y \Rrightarrow R & (\Rrightarrow +) \\ \hline X,Y \vdash Y \Rrightarrow R & (\mathrm{Thin_e}) \\ \hline (X,Y);Y \vdash R & (\Rrightarrow -) \\ \hline Y \vdash (X,Y) \Rrightarrow R & (\mathrm{Comm_i})\text{ and }(\Rrightarrow +) \\ \hline X,Y \vdash (X,Y) \Rrightarrow R & (\mathrm{Thin_e})\text{ and }(\mathrm{Comm_e}) \\ \hline (X,Y);(X,Y) \vdash R & (\Rrightarrow -) \\ \hline X,Y \vdash R & (\mathrm{Contr_i}) \end{array}$$

[5]Since unlike Read and Slaney we do not only use the comma and the semicolon as binary operators but combine this idea with defining extensional and intensional rules of higher levels, it is not necessary for us to allow for the inference schemata above to operate on "subbunches" of nested bunches in antecedents. For example, it follows from our inference schemata that (X,X) can be replaced by X not only as a full antecedent of a sequent but also as a part of a nested bunch. E.g., from $((X,X);Y),Z \vdash A$ follows by $(\Rightarrow +)$ and $(\Rrightarrow +)$ the sequent $X,X \vdash Y \Rrightarrow (Z \Rightarrow A)$, thus by application of $(\mathrm{Contr_e})$ $X \vdash Y \Rrightarrow (Z \Rightarrow A)$, and by $(\Rightarrow -)$ and $(\Rrightarrow -)$ we obtain $(X;Y),Z \vdash A$.

$$\text{(iii)}\quad \begin{array}{c} \emptyset \vdash X \Rightarrow R \\ \hline\hline \emptyset , X \vdash R \\ \hline\hline X \vdash R \\ \hline\hline \Upsilon ; X \vdash R \\ \hline\hline \Upsilon \vdash X \Rrightarrow R \end{array} \begin{array}{l} \\ (\Rightarrow) \\ (\mathrm{Id_e}) \\ (\mathrm{Id_i}) \\ (\Rrightarrow) \\ \end{array} \qquad\qquad \text{(iv)}\quad \begin{array}{c} \emptyset \vdash X \Rrightarrow R \\ \hline\hline \emptyset ; X \vdash R \\ \hline\hline \Upsilon ; (\emptyset ; X) \vdash R \\ \hline\hline \Upsilon \vdash (\emptyset ; X) \Rrightarrow R \end{array} \begin{array}{l} \\ (\Rrightarrow) \\ (\mathrm{Id_i}) \\ (\Rrightarrow) \\ \end{array}$$

(v), (vi), (vii) similarly. □

Lemma 5.2 All sequents of the following form are derivable:
(i) $R \dashv\vdash \emptyset \Rightarrow R$ (ii) $R \dashv\vdash \Upsilon \Rrightarrow R$ (iii) $X \Rrightarrow R \vdash X \Rightarrow R$.

Proof (i) follows by use of $(\Rightarrow)$ and $(\mathrm{Id_e})$. (ii) follows by use of $(\Rrightarrow)$ and $(\mathrm{Id_i})$. (iii) follows by use of $(\Rrightarrow -)$ and $(\Rightarrow +)$ from Lemma 5.1 (ii). □

Since we have two different empty bunches $\emptyset$ and Υ, we must distinguish between $\emptyset \vdash R$ and $\Upsilon \vdash R$ (thus "$\vdash R$" is ambiguous). The meaning of $\emptyset \vdash R$ can be circumscribed as: "R follows from *any* assumption", whereas $\Upsilon \vdash R$ means: "R follows without assumption". This difference makes some considerations necessary of how to formulate the inference schema which introduces primitive rules into a derivation. In SC3+ it took the form

$$R \,\frac{}{\emptyset \vdash R}\,(\text{Prim})\ .$$

We could take this inference schema over to the present system by interpreting $\emptyset$ as a bunch rather than a set. However, another possibility would be to choose the schema

$$R \,\frac{}{\Upsilon \vdash R}\,(\Upsilon\text{-Prim})\ ,$$

which according to Lemma 5.1 (i) is weaker than (Prim).

It can be shown that both alternatives are equivalent in the sense that the resulting systems can be embedded into each other. If we interpret every primitive rule of the form $X \Rightarrow R$ of the system with (Prim) as $X \Rrightarrow R$ in the system with (Υ-Prim) and every primitive rule of the form $X \Rrightarrow R$ as $(\emptyset ; X) \Rrightarrow R$, then from Lemma 5.1 ((iii) and (iv)) follows that in the resulting systems the same sequents are derivable. Conversely, if we interpret every primitive rule of the form $X \Rrightarrow R$ of the system with (Υ-Prim) as $X \Rightarrow R$ in the system with (Prim) and each primitive rule of the form $X \Rightarrow R$ as $(\Upsilon , X) \Rightarrow R$, then from Lemma 5.1 ((iii) and (v)) follows that in the resulting systems the same sequents are derivable. (Primitive rules of the form A are in all cases translated into themselves.)

Furthermore, if we choose the system with (Prim), we can restrict ourselves to extensional primitive rules, since intensional primitive rules can be expressed by extensional ones (Lemma 5.1 (vi)). Correspondingly, if we choose the system with (Υ-Prim), we can restrict ourselves to intensional primitive rules, since extensional primitive rules can be expressed by intensional ones (Lemma 5.1 (vii)). For our system SCR we take (Prim) and require that primitive rules be extensional. Table 4 presents all inference schemata of SCR.

TABLE 4
Inference schemata of the structural systems SCR

$$\begin{array}{c} X,(Y,Z)\vdash R \\ \hline\hline (X,Y),Z\vdash R \end{array}\,(\mathrm{Assoc_e}) \qquad \begin{array}{c} X;(Y;Z)\vdash R \\ \hline\hline (X;Y);Z\vdash R \end{array}\,(\mathrm{Assoc_i})$$

$$\begin{array}{c} X,Y\vdash R \\ \hline\hline Y,X\vdash R \end{array}\,(\mathrm{Comm_e}) \qquad \begin{array}{c} X;Y\vdash R \\ \hline\hline Y;X\vdash R \end{array}\,(\mathrm{Comm_i})$$

$$\begin{array}{c} X,X\vdash R \\ \hline\hline X\vdash R \end{array}\,(\mathrm{Contr_e}) \qquad \begin{array}{c} X;X\vdash R \\ \hline\hline X\vdash R \end{array}\,(\mathrm{Contr_i})$$

$$\begin{array}{c} \emptyset,X\vdash R \\ \hline\hline X\vdash R \end{array}\,(\mathrm{Id_e}) \qquad \begin{array}{c} \Upsilon;X\vdash R \\ \hline\hline X\vdash R \end{array}\,(\mathrm{Id_i})$$

$$\begin{array}{c} X\vdash R \\ \hline X,Y\vdash R \end{array}\,(\mathrm{Thin_e})$$

$$\begin{array}{c} \\ \hline R\vdash R \end{array}\,(\mathrm{Refl}) \qquad R\,\begin{array}{c} \\ \hline \emptyset\vdash R \end{array}\,(\mathrm{Prim}) \quad (R \text{ extensional primitive rule})$$

$$\begin{array}{c} X\vdash R \quad R\vdash R' \\ \hline X\vdash R' \end{array}\,(\mathrm{Trans})$$

$$\begin{array}{c} X,Y\vdash R \\ \hline\hline X\vdash Y\Rightarrow R \end{array}\,(\Rightarrow) \qquad \begin{array}{c} X;Y\vdash R \\ \hline\hline X\vdash Y\Rrightarrow R \end{array}\,(\Rrightarrow)$$

The system SCR is obviously an extension of the system SC3+ in the following sense: If we restrict ourselves to bunches built up by means of $\emptyset$, the comma and the double arrow alone, we can interpret bunches as finite sets, where $\emptyset$ is read as the empty set. The inference schemata (Assoc$_e$), (Comm$_e$), (Contr$_e$) and (Id$_e$) become superfluous, and we obtain exactly the schemata of SC3+.

In order to prove a replacement theorem, we need the notion of a *subbunch*:

Every bunch is a subbunch of itself.
Every subbunch of X and Y is a subbunch of (X,Y) and $(X;Y)$.
Every subbunch of X and R is a subbunch of $(X\Rightarrow R)$ and $(X\Rrightarrow R)$.
By $Z[X]$ we denote a bunch in which X occurs at a certain place as a subbunch. If used in the same context, $Z[Y]$ then denotes the result of replacing this occurrence of X in $Z[X]$

by Y, provided this result is a bunch (e.g., a replacement of the subbunch R of $X \Rightarrow R$ by a bunch of the form $(Y;Z)$ is not defined).

Furthermore, we define $X \Vdash Y$ as follows:
$X \Vdash Y$ means that the inference schema

$$\frac{Y \vdash R}{X \vdash R}$$

is admissible, i.e., if for every R the derivability of $Y \Rightarrow R$ in SCR implies that of $X \Rightarrow R$. As a limiting case we have that $X \Vdash R$ means the same as "$X \Rightarrow R$ is derivable". $X \dashv\Vdash Y$ expresses that $X \Vdash Y$ and $Y \Vdash X$.

Theorem 5.3 (Replacement Theorem) If $X \dashv\Vdash Y$, then $Z[X] \dashv\Vdash Z[Y]$, provided the replacement of X by Y in $Z[X]$ is defined.

Proof by induction on the definition of subbunches. If $Z[X]$ is X, then the assertion is trivial. Suppose $Z[X]$ is $(Z_1[X], Z_2)$. Then $Z_1[X] \dashv\Vdash Z_1[Y]$ by induction hypothesis. Now suppose $Z_1[Y], Z_2 \vdash R$. Then $Z_1[Y] \vdash Z_2 \Rightarrow R$, hence $Z_1[X] \vdash Z_2 \Rightarrow R$, i.e., $Z_1[X], Z_2 \vdash R$. Thus we have $Z[X] \Vdash Z[Y]$. Analogously we show $Z[Y] \Vdash Z[X]$. The same holds for the case where $Z[X]$ is $(Z_1, Z_2[X])$ or $(Z_1[X]; Z_2)$ or $(Z_1; Z_2[X])$. Suppose $Z[X]$ is $Z_1 \Rightarrow R'[X]$. Then $Z_1 \Rightarrow R'[X] \dashv\vdash Z_1 \Rightarrow R'[Y]$ by induction hypothesis and (Trans), therefore $Z[X] \dashv\Vdash Z[Y]$ by (Trans). Suppose $Z[X]$ is $Z_1[X] \Rightarrow R'$. Then $Z_1[X] \vdash (Z_1[X] \Rightarrow R') \Rightarrow R'$, i.e., $Z_1[X] \Rightarrow R' \vdash Z_1[Y] \Rightarrow R'$, and similarly the converse. If $Z[X]$ is $Z_1[X] \Rrightarrow R'$, we proceed analogously. $\square$

§ 2. Logical constants of positive relevance logic

In this section we carry over the ideas of Ch. 2, restricted to the propositional case, to our more complicated structural apparatus involving bunches of higher-level rules.

Formulae are considered to be either *atomic formulae* (the form of which can be left open) and expressions of the form $S(A_1, \ldots, A_n)$ for formulae $A_1, \ldots, A_n$, if S is an n-ary logical constant (also called connective). An n-ary *formula schema* is defined as in Ch. 2 § 2 by use use of schematic letters $p_1, \ldots, p_n$ for formulae: If $i \leq n$, then p_i is an n-ary formula schema. If $\alpha_1, \ldots, \alpha_k$ are n-ary formula schemata and S is a k-ary connective, then $S(\alpha_1, \ldots, \alpha_k)$ is an n-ary formula schema.

Rule schemata and *schemata of bunches* are then defined as rules and bunches with formula schemata playing the role of formulae: Every n-ary formula schema is an n-ary rule schema. $\emptyset$ and Υ are n-ary schemata of bunches. Every n-ary rule schema is an n-ary schema of a bunch. If Δ_1 and Δ_2 are n-ary schemata of bunches, then so are (Δ_1, Δ_2) and $(\Delta_1; \Delta_2)$. If Δ is an n-ary schema of a bunch and ϱ an n-ary rule schema, then $\Delta \Rightarrow \varrho$ and $\Delta \Rrightarrow \varrho$ are n-ary rule schemata. Schemata of bunches correspond to the sets of rule schemata of Ch. 2. The notational conventions of Ch. 2 remain in force, particularly those concerning the notations $\Delta_i[\underline{p}]$, $\Delta_i[\underline{A}]$, $S(\underline{p})$, $S(\underline{A})$ etc.

An *assertibility condition* for an n-ary logical constant is an n-ary schema of a bunch. With each n-ary connective S, a *nonempty* set

$$\mathbb{D}(S) = \{\Delta_1[p_1, \ldots, p_n], \ldots, \Delta_m[p_1, \ldots, p_n]\}$$

of assertibility conditions for S is assumed to be associated. We require $\mathbb{D}(S)$ to be nonempty, because the justification of the "ex falso quodlibet" as based on an empty set of assertibility conditions is problematic in relevance logic. In the next section, we shall introduce intensional absurdity and negation by corresponding structural concepts. Here we only discuss the positive fragment.

It is again required that the logical constants considered can be ordered in a sequence

$$S_1, S_2, \ldots, S_i, \ldots$$

in such a way that for any i, $\mathbb{D}[S_i]$ only contains logical constants S_j for $j < i$.

If $\mathbb{D}(S) = \{\Delta_1[\underline{p}], \ldots, \Delta_m[\underline{p}]\}$, then the rule schemata of S-introduction are

$$\begin{array}{l} \Delta_1[\underline{p}] \Rightarrow S(\underline{p}) \\ \quad\vdots \\ \Delta_m[\underline{p}] \Rightarrow S(\underline{p}) \ , \end{array} \qquad (S\text{-I})$$

and the schema of S-elimination is

$$(S(\underline{p}), (\Delta_1[\underline{p}] \Rightarrow r), \ldots, (\Delta_m[\underline{p}] \Rightarrow r)) \Rightarrow r \ , \qquad (S\text{-E})$$

where r is a schematic letter which may be instantiated by *rules* and not only by formulae. (Strictly speaking, (S-E) is not a rule schema in the sense defined, since in rule schemata only schematic letters for formulae and not for rules are allowed to occur—see footnote 1 of Ch. 3.)

With the exception of the schematic letter r for rules in (S-E), the schemata for logical rules look exactly as those given in Ch. 2 § 2. This is not counterintuitive. The case of relevance logic differs from that of "ordinary" logic in that the structural framework is more sophisticated. The general way logical constants are characterized is independent of that difference. Of course, the assertibility conditions $\Delta_i[\underline{p}]$ of a constant S may have a richer structure since they are now schemata of bunches. But this difference does not affect the general pattern of introduction and elimination rules. If one looks at the "common content"-motivation of logical rules (Theorem 2.4), there is in fact no reason why in relevance logic the general pattern of logical rules should differ from the case discussed in Ch. 2. The condition that for all $\underline{A}$ and all R, $S(\underline{A})$ expresses the common content of $\Delta_1[\underline{A}], \ldots, \Delta_m[\underline{A}]$, i.e.,

$$S(\underline{A}) \vdash R \quad \text{iff} \quad \text{for each } i \ (1 \leq i \leq m) \ \Delta_i[\underline{A}] \vdash R \ ,$$

is completely independent of whether we are dealing with bunches or not. As can easily be seen, the proof of a theorem corresponding to Theorem 2.4 (restricted to the propositional case) would only use laws for $\emptyset$, the comma and the double arrow "$\Rightarrow$".

However, if one would like to have a rule schema for S-elimination that uses intensional combinations, one may take

$$(S(\underline{p}); ((\Delta_1[\underline{p}] \Rrightarrow r), \ldots, (\Delta_m[\underline{p}] \Rrightarrow r))) \Rightarrow r \ .$$

It can be shown that this schema and (S-E) are equivalent as schemata for primitive rules.

A replacement theorem for subformulae corresponding to Theorem 2.3 can be proved in the same way as in Ch. 2, now based on Theorem 5.3. Furthermore, a separation theorem like Theorem 2.5 is also available in the present case.[6]

The standard logical constants of positive relevance logic which can be characterized by our general schemata are extensional conjunction (&), disjunction ($\vee$), extensional implication ($\supset$), extensional truth ($\top$), intensional truth ($\mathtt{t}$), intensional implication or fusion ($\circ$) and intensional or relevant implication ($\rightarrow$). Their assertibility conditions and primitive rules are given in Table 5. The constants which are new are the nullary constant $\mathtt{t}$ (corresponding to $\curlyvee$), fusion $\circ$ (corresponding to the semicolon) and relevant implication $\rightarrow$ (corresponding to the triple arrow). Since we here only deal with positive logic, we need the nullary extensional constant $\top$, which in the non-relevant case was definable as $\bot \supset \bot$. Since rule schemata are themselves schemata of bunches, $p \Rightarrow q$ is now an assertibility condition for $\supset$. In the non-relevant case, assertibility conditions were always sets of rule schemata, so that only $\{p \Rightarrow q\}$ was an assertibility condition for $\supset$. The analogue applies to other connectives.

There is no relevant analogue to disjunction $\vee$. This is because disjunction has no immediate relation to the structural framework but only to the way logical constants are defined: Disjunction is the only standard constant S with $\mathbb{D}(S)$ having more than one element. In the literature on relevance logic there is normally an intensional version of disjunction being considered, which is sometimes called "fission" and denoted by "+" (see Anderson and Belnap 1975, p. 344; Routley et al. 1982, p. 361). It is usually defined as

$$A + B =_{\mathrm{df}} \sim A \rightarrow B$$

or

$$A + B =_{\mathrm{df}} \sim(\sim A \circ \sim B) \ ,$$

where $\sim$ is a negation which behaves like classical Boolean negation (and which, incidentally, makes fusion $\circ$ and relevant implication $\rightarrow$ interdefinable). Such a negation will be considered in the next section. In our framework there seems to be no way of characterizing fission independently, i.e. without reference to a constant of classical absurdity or negation.

As remarked in Table 5, for $\rightarrow$-elimination there is an equivalent version available, which is a kind of intensional modus ponens:

$$(p \rightarrow q; p) \Rightarrow q \ .$$

[6] As in Ch. 2, we do not present the proof of this result here, since it is not directly connected with our central questions of structural frameworks. Although somewhat lengthy, it uses the standard methods of normalization proofs.

TABLE 5
Standard constants of positive relevance logic and their primitive rules

(Binary connectives are written in infix notation. The letters p and q stand for p_1 and p_2, respectively. The letter r is a schematic letter to be instantiated by rules.)

$S(\underline{p})$	$\mathbb{D}(S)$	Schemata for I rules	Schemata for E rules
$p \,\&\, q$	$\{\{p, q\}\}$	$p, q \Rightarrow p \,\&\, q$	$(p \,\&\, q, ((p, q) \Rightarrow r)) \Rightarrow r$ equivalent: $p \,\&\, q \Rightarrow p$ $p \,\&\, q \Rightarrow q$
$p \vee q$	$\{p, q\}$	$p \Rightarrow p \vee q$ $q \Rightarrow p \vee q$	$(p \vee q, (p \Rightarrow r), (q \Rightarrow r)) \Rightarrow r$
$p \supset q$	$\{p \Rightarrow q\}$	$(p \Rightarrow q) \Rightarrow p \supset q$	$(p \supset q, ((p \Rightarrow q) \Rightarrow r)) \Rightarrow r$ equivalent: $p \supset q, p \Rightarrow q$
$\top$	$\{\emptyset\}$	$\emptyset \Rightarrow \top$	$(\top, (\emptyset \Rightarrow r)) \Rightarrow r$ (derivable)
$\mathtt{t}$	$\{\curlyvee\}$	$\curlyvee \Rightarrow \mathtt{t}$	$(\mathtt{t}, (\curlyvee \Rightarrow r)) \Rightarrow r$ (derivable)
$p \circ q$	$\{\{p; q\}\}$	$(p \circ q) \Rightarrow p \circ q$	$(p \circ q, ((p; q) \Rightarrow r)) \Rightarrow r$
$p \to q$	$\{p \Rrightarrow q\}$	$(p \Rrightarrow q) \Rightarrow p \to q$	$(p \to q, ((p \Rrightarrow q) \Rightarrow r)) \Rightarrow r$ equivalent: $p \to q; p \Rightarrow q$

For $\circ$-elimination, there is no version available, which corresponds to the "direct" elimination rules

$$p \,\&\, q \Rightarrow p \qquad\qquad p \,\&\, p \Rightarrow q$$

for extensional conjunction.

The functional completeness of the standard constants $\&$, $\vee$, $\supset$, $\top$, $\mathtt{t}$, $\circ$ and $\to$ is obtained by using the following mapping of schemata of bunches to formula schemata:

$g(\alpha) = \alpha$
$g(\emptyset) = \top$
$g(\curlyvee) = \mathtt{t}$

$g((\Delta_1, \Delta_2)) = g(\Delta_1) \,\&\, g(\Delta_2)$
$g((\Delta_1; \Delta_2)) = g(\Delta_1) \circ g(\Delta_2)$
$g(\Delta \Rightarrow \varrho) = g(\Delta) \supset g(\varrho)$
$g(\Delta \Rightarrow \varrho) = g(\Delta) \rightarrow g(\varrho)$.

Complex formula schemata can then be translated as follows:

$$h(S(\underline{p})) = g(\Delta_1[\underline{p}]) \vee \ldots \vee g(\Delta_m[\underline{p}]), \text{ if } \mathbb{D}(S) = \{\Delta_1[\underline{p}], \ldots, \Delta_m[\underline{p}]\} \ .$$

Then it can be shown that $S(\underline{p})$ and $h(S(\underline{p}))$ are schematically equivalent. From this result we obtain, by using replacement of subformulae and separativity of logical constants, that $X \vdash R$ is derivable in a system with an arbitrary number of logical constants iff $X^\circ \vdash R^\circ$ is derivable in a system with only the standard connectives, whereby X° and R° result from X and R, respectively, by replacing complex formulae according to the function h.

§ 3. Absurdity and negation in relevance logic

A central motive for relevance logic is that one is interested in which assumptions are really needed if one wants to establish an assertion under assumptions. In our framework this idea is captured by using a way of combining assumptions for which thinning is not allowed. Another basic issue of relevance logic, inasmuch as it goes beyond positive logics and deals with absurdity and negation, is the rejection of the "ex falso quodlibet". The reason for this rejection is that neither absurdity (as a primitive constant) nor a contradiction is related in any "relevant" sense to arbitrary formulae.

Therefore in relevance logic no "natural" notion of logical absurdity is available, obtained as in (non-relevant) intuitionistic logic by considering an empty set of assertibility conditions and leading to the "ex falso quodlibet" as a primitive rule. In order to obtain notions of absurdity and negation for relevance logic we propose to enlarge the structural framework, following the ideas sketched in Ch. 3. Since we have already $\emptyset$ and $\curlyvee$ as extensional and intensional objects expressing truth, we introduce corresponding extensional and intensional objects $\#$ and $\curlywedge$ (respectively) expressing absurdity. The modified definition of rules and bunches is then obtained from the original one by considering in addition $\#$ and $\curlywedge$ as rules (and therefore bunches) of level 0, where $\#$ is an extensional and $\curlywedge$ an intensional rule.

Correspondingly, on the logical side we obtain two additional constants F and f, with $\mathbb{D}(\mathsf{F}) = \{\#\}$ and $\mathbb{D}(\mathsf{f}) = \{\curlywedge\}$, i.e., with primitive rules according to the schemata

$$\# \rightarrow \mathsf{F} \tag{F-I}$$

$$(\mathsf{F}, (\# \Rightarrow r)) \Rightarrow r \ , \text{ equivalently: } \mathsf{F} \Rightarrow \# \tag{F-E}$$

and

$$\curlywedge \Rightarrow \mathsf{f} \tag{f-I}$$

$$(\mathsf{f}\,,(\curlywedge \Rightarrow r)) \Rightarrow r \text{ , equivalently: } \mathsf{f} \Rightarrow \curlywedge \text{ .} \qquad (\mathsf{f}\text{-E})$$

Extensional negation ("$\neg$") and intensional negation ("$\sim$") can then be characterized by

$$(p \Rightarrow \#) \Rightarrow \neg p \qquad (\neg\text{-I})$$

$$(\neg p, ((p \Rightarrow \#) \Rightarrow r)) \Rightarrow r \text{ , equivalently: } \neg p, p \Rightarrow \# \qquad (\neg\text{-E})$$

and

$$(p \Rrightarrow \curlywedge) \Rightarrow {\sim} p \qquad ({\sim}\text{-I})$$

$$({\sim} p, ((p \Rrightarrow \curlywedge) \Rightarrow r)) \Rightarrow r \text{ , equivalently: } {\sim} p ; p \Rightarrow \curlywedge \text{ .} \qquad ({\sim}\text{-E})$$

Another possibility for defining $\neg$ and $\sim$ would be to use two sorts of structural negation (e.g. denoted by "$-$" and "$\backslash$"), and then to use logical rules according to which $\neg$ expresses $-$ and $\sim$ expresses $\backslash$ (see Ch. 3). We follow the approach which only uses absurdity as basic.

The crucial question is which additional inference schemata should be given for # and $\curlywedge$. The simplest way is to add *no* inference schema. The resulting system may be called *minimal relevance logic*, since the laws we obtain for the extensional part (i.e. for the formulae formulated by use of &, $\vee$, $\supset$ and F) correspond to those of minimal logic. In particular, we can now derive $\emptyset \vdash \mathsf{F} \supset \neg A$ for any A. (For this derivation thinning is essential. So we cannot derive the intensional analogue $\emptyset \vdash \mathsf{f} \rightarrow {\sim} A$.)

To allow for an "ex falso quodlibet" for intensional absurdity would contradict the philosophical essentials of relevance logic. But one might perhaps use it in connection with #, since # is considered *extensional* absurdity for which the traditional laws may hold. This leads to adding the inference schema

$$\frac{X \vdash \#}{X \vdash R}\,(\mathrm{EFQ}_{\#})$$

to the structural framework (whereas $\curlywedge$ still has no characteristic inference schema). Such a system may be called "intuitionistic relevance logic", since the typical intuitionistic law of ex falso quodlibet holds for extensional absurdity, without any typical classical law (such as classical reductio ad absurdum), holding in the system.

The usual way of presenting relevance logic, however, is to treat it in a classical setting, where "classical" means "classical with respect to the intensional operations". In our framework, we can express this by postulating

$$\frac{X ; (R \Rrightarrow \curlywedge) \vdash \curlywedge}{X \vdash R}\,(\curlywedge_{\mathrm{c}})$$

as an inference schema. The system SCR, extended by ($\curlywedge_{\mathrm{c}}$) may be called a "structural framework for classical relevance logic". We denote it by $\mathrm{SCR}_{\mathrm{c}}$.

In $\mathrm{SCR_c}$ we can derive $\emptyset \Rrightarrow \curlywedge \vdash R$ for any rule R:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \Rrightarrow \curlywedge \vdash \emptyset \Rrightarrow \curlywedge}\,(\mathrm{Refl})}{(\emptyset \Rrightarrow \curlywedge); \emptyset \vdash \curlywedge}\,(\Rrightarrow -)}{\emptyset \vdash (\emptyset \Rrightarrow \curlywedge) \Rrightarrow \curlywedge}\,(\mathrm{Comm_i}) \text{ and } (\Rrightarrow +)}{R \Rrightarrow \curlywedge \vdash (\emptyset \Rrightarrow \curlywedge) \Rrightarrow \curlywedge}\,(\mathrm{Thin_e}) \text{ and } (\mathrm{Id_e})}{(\emptyset \Rrightarrow \curlywedge); (R \Rrightarrow \curlywedge) \vdash \curlywedge}\,(\Rrightarrow -) \text{ and } (\mathrm{Comm_i})}{\emptyset \Rrightarrow \curlywedge \vdash R}\,(\curlywedge_{\mathrm{c}}) \ .$$

This means that if we *define* # by $\emptyset \Rrightarrow \curlywedge$, then we obtain ($\mathrm{EFQ}_{\#}$) as a derived inference schema. However, we conjecture that one *cannot* derive in $\mathrm{SCR_c}$

$$(R \Rightarrow (\emptyset \Rrightarrow \curlywedge)) \Rightarrow (\emptyset \Rrightarrow \curlywedge) \vdash R \ .$$

If this conjecture is right, then one obtains an even stronger system than $\mathrm{SCR_c}$ by requiring in addition to the classical law ($\curlywedge_{\mathrm{c}}$) for $\curlywedge$ and $\Rrightarrow$ the corresponding classical law for # and $\Rightarrow$:

$$\dfrac{X, (R \Rrightarrow \#) \vdash \#}{X \vdash R}\,(\#_{\mathrm{c}}) \ .$$

However, the standard system R of relevance logic comes closest to $\mathrm{SCR_c}$ with # being defined as $\emptyset \Rrightarrow \curlywedge$.

§ 4. The Read-Slaney calculus for relevance logic

In our general formalism relevance logic with arbitrary many logical operators can be treated in such a way that the rules for these operators follow a uniform pattern. Rules for the specific constants T, F, &, $\supset$, $\vee$, t, f, $\circ$ and $\rightarrow$ are special instances of this pattern. These standard connectives are complete in that they suffice to express everything that can be expressed by use of other logical constants. Our system has been developed for theoretical purposes, aiming at a general framework for relevance logic, with respect to which such general results as functional completeness can be obtained. For the practical purpose of carrying out derivations in relevance logic with the above standard connectives, it is convenient to use a simpler system, even if this means to give up certain theoretical principles which were essential for the construction of our general structural framework.

As such a system we present a slightly modified version, called "RS", of calculi developed by Read and Slaney (see footnote 1). The system RS differs from our general system (restricted to primitive rules for the above standard connectives) in that no apparatus of higher-level rules is employed, and that logical rules are identified with certain specific inference schemata (rather than considering rules as separate objects to be imported into derivations by a schema like (Prim)). Nevertheless, in using the bunch structure at least for formulae and arranging logical inference schemata according to the introduction-elimination pattern, the system RS is conceptually still closely related to our framework. We show that every sequent which can be derived in RS can also be derived in $\mathrm{SCR_c}$, and conversely, that every sequent which can be expressed in RS and derived in $\mathrm{SCR_c}$, can also be derived in RS. This demonstrates that our system is in fact a generalization for theoretical purposes of Read's and Slaney's systems, which are very easy to handle and are particularly well-suited for teaching purposes.

By $\mathrm{SCR_c}$ we now understand the system defined in the previous section with only $\curlywedge$ as (intensional) structural absurdity (since $\#$ is definable as $\emptyset \Rrightarrow \curlywedge$) and only introduction and elimination rules for the standard connectives T, &, $\supset$, $\vee$, t, f, $\circ$ and $\rightarrow$ (since we do not have $\#$ as primitive, we leave out F).

RS is a system whose formulae are built up from the same standard connectives as $\mathrm{SCR_c}$. RS-bunches are defined as follows:

Every formula is an RS-bunch. If X and Y are RS-bunches, then so are (X, Y) and $(X; Y)$.

RS-subbunches are defined as follows: Every RS-bunch is an RS-subbunch of itself. Every RS-subbunch of X or Y is an RS-subbunch of (X, Y) and $(X; Y)$. As notation for an RS-bunch in which an RS-subbunch occurs at a certain place we use $Z[X]$.

In RS-bunches, the roles of $\emptyset$, $\curlyvee$ and $\curlywedge$ are taken over by the *formulae* T, t and f, respectively. Although from the theoretical viewpoint this means that the structural and the logical levels are confounded, this procedure shortens derivations considerably. Applications of the trivial introduction and elimination rules for T, t and f can simply be dropped.

In RS, rules as parts of bunches are not defined. The notion of a bunch of rules can be avoided when dealing with the standard connectives rather than with arbitrary n-ary operators. The price paid for this is that for extensional and intensional implication the standardized elimination rules can no longer be formulated, but only their equivalent versions of extensional and intensional modus ponens.

An RS-sequent is of the form $X \vdash A$ where X is an RS-bunch and A a formula. The formalism RS has the inference schemata presented in Table 6, where $X \geq Y$ is used as an abbreviation for

$$\frac{U[X] \vdash A}{U[Y] \vdash A}\ ,$$

and $X = Y$ stands for $X \geq Y$ and $Y \geq X$.

Obviously, each RS-sequent $X \vdash A$ is also a sequent in the sense of $\mathrm{SCR_c}$.

Theorem 5.4 If $X \vdash A$ is derivable in RS, then $X \vdash A$ is derivable in $\mathrm{SCR_c}$.

Proof We show that all inference schemata of RS are admissible in $\mathrm{SCR_c}$. Consider first the schemata for the manipulation of RS-subbunches, which have the general form:

$$\frac{U[X] \vdash A}{U[Y] \vdash A}\ .$$

TABLE 6
Inference schemata of RS

Manipulation of RS-subbunches

$X, (Y, Z) = (X, Y), Z$	$X; (Y; Z) = (X; Y); Z$
$X, Y = Y, X$	$X; Y = Y; X$
$X, X = X$	$X; X = X$
$\mathsf{T}, X = X$	$\mathsf{t}; X = X$
$X \geq X, Y$	

Schema for introducing assumptions

$$\frac{}{A \vdash A}\,(\text{Assum})$$

Classical absurdity schema

$$\frac{X; (A \to \mathsf{f}) \vdash \mathsf{f}}{X \vdash A}\,(\mathrm{f_c})$$

Introduction and elimination schemata

$$\frac{X \vdash A \quad Y \vdash B}{X, Y \vdash A \,\&\, B} \qquad \frac{X \vdash A \,\&\, B}{X \vdash A} \qquad \frac{X \vdash A \,\&\, B}{X \vdash B}$$

$$\frac{X \vdash A \quad Y \vdash B}{X; Y \vdash A \circ B} \qquad \frac{X \vdash A \circ B \quad Y[A; B] \vdash C}{Y[X] \vdash C}$$

$$\frac{X \vdash A}{X \vdash A \vee B} \qquad \frac{X \vdash B}{X \vdash A \vee B} \qquad \frac{X \vdash A \vee B \quad Y[A] \vdash C \quad Y[B] \vdash C}{Y[X] \vdash C}$$

$$\frac{X, A \vdash B}{X \vdash A \supset B} \qquad \frac{X \vdash A \supset B \quad Y \vdash A}{X, Y \vdash B}$$

$$\frac{X; A \vdash B}{X \vdash A \to B} \qquad \frac{X \vdash A \to B \quad Y \vdash A}{X; Y \vdash B}$$

By applying ($\Rightarrow$) and ($\Rrightarrow$), we can construct derivations

$$\begin{array}{c} U[X] \vdash A \\ \vdots \\ X \vdash R \end{array} \qquad\qquad \begin{array}{c} Y \vdash R \\ \vdots \\ U[Y] \vdash A \end{array}$$

in SCR$_\mathrm{c}$ for a certain rule R, i.e., we can “unpack” the RS-subbunches X and Y from the “surrounding” RS-bunch $U[\,]$. Now the inference step

$$\frac{X\vdash R}{Y\vdash R}$$

is available in SCR$_\mathrm{c}$, if we use that in SCR$_\mathrm{c}$ we can derive $\mathsf{T}\dashv\!\Vert\!\vdash\emptyset$ and $\mathsf{t}\dashv\!\Vert\!\vdash\curlyvee$.
The schema (Assum) of RS is a special case of the schema (Refl) of SCR$_\mathrm{c}$. Since in SCR$_\mathrm{c}$ we have $A\rightarrow\mathsf{f}\dashv\!\Vert\!\vdash A\Rrightarrow\mathsf{F}$, ($\mathsf{f}_\mathrm{c}$) can be shown to be admissible in SCR$_\mathrm{c}$ by use of ($\curlywedge_\mathrm{c}$). Concerning the introduction and elimination schemata, we consider as examples $\circ$-introduction and $\circ$-elimination. For $\circ$-introduction, take the following derivation in SCR$_\mathrm{c}$:

$$\dfrac{Y\vdash B \qquad \dfrac{\dfrac{\dfrac{X\vdash A \qquad \dfrac{\dfrac{(\circ\text{-I})\;\dfrac{}{\emptyset\vdash(A;B)\Rightarrow A\circ B}\,\text{(Prim)}}{A;B\vdash A\circ B}\,(\Rightarrow -)\text{ and (Contr}_\mathrm{e}\text{)}}{A\vdash B\Rrightarrow A\circ B}\,(\Rrightarrow +)}{X\vdash B\Rrightarrow A\circ B}\,\text{(Trans)}}{X;B\vdash A\circ B}\,(\Rrightarrow -)}{B\vdash X\Rrightarrow A\circ B}\,\text{(Comm}_\mathrm{i}\text{) and }(\Rrightarrow +)}{\dfrac{Y\vdash X\Rrightarrow A\circ B}{X;Y\vdash A\circ B}\,(\Rrightarrow -)\text{ and (Comm}_\mathrm{i}\text{)}}\,\text{(Trans)}\ .$$

For $\circ$-elimination we do not write down the derivation in SCR$_\mathrm{c}$ but just sketch how we have to proceed. From $Y[A;B]\vdash C$ we obtain, by applying $(\Rightarrow +)$ and $(\Rrightarrow +)$, $\emptyset\vdash(A;B)\Rightarrow R$ for a certain R. Furthermore, from the rule of $\circ$-elimination in SCR$_\mathrm{c}$ in the form $(A\circ B,((A;B)\Rightarrow R))\Rightarrow R$ and from $X\vdash A\circ B$ we obtain $(A;B)\Rightarrow R\vdash X\Rightarrow R$, hence by (Trans) $\emptyset\vdash X\Rightarrow R$. Applications of $(\Rightarrow -)$ and $(\Rrightarrow -)$ then yield $Y[X]\vdash C$. □

The converse to this theorem should be formulated as follows: For all RS-bunches X and formulae A, if $X\vdash A$ can be derived in RS then $X\vdash A$ can be derived in SCR$_\mathrm{c}$. This means that SCR$_\mathrm{c}$ is “conservative” in a certain sense in regard to RS, i.e., it does not permit us to derive more as far as matters are concerned which can be expressed in RS.

We first define a translation * of bunches into RS-bunches. Here the mapping g is taken from § 2 of this chapter, but understood as a mapping from bunches to formulae, and not only from schemata of bunches to formula schemata.

$R^*=g(R)$ for rules R
$\emptyset^*=\mathsf{T}$
$\curlywedge^*=\mathsf{t}$
$\curlyvee^*=\mathsf{f}$
$(X,Y)^*=(X^*,Y^*)$
$(X;Y)^*=(X^*;Y^*)$.

Obviously, X^* is always an RS-bunch. It is easy to see that X^* is X, if X is an RS-bunch.

Theorem 5.5 If $X \vdash R$ is derivable in $\mathrm{SCR_c}$, then $X^* \vdash R^*$ is derivable in RS.

Proof We show that if

$$\frac{X_1 \vdash R_1 \ \ldots \ X_n \vdash R_n}{X \vdash R}$$

is an inference schema of $\mathrm{SCR_c}$, then

$$\frac{X_1^* \vdash R_1^* \ \ldots \ X_n^* \vdash R_n^*}{X^* \vdash R^*}$$

is admissible in RS. For the inference schemata which only manipulate the structure of antecedents as well as for (Assum) this is obvious. Concerning (Trans), we can easily show the admissibility of

$$\frac{X^* \vdash R^* \ \ldots \ R^* \vdash R'^*}{X^* \vdash R'^*}$$

in RS by first applying →-introduction to the right upper sequent and then using →-elimination. Since $X^*;(R \Rrightarrow \curlywedge)^* \vdash \curlywedge^*$ is the same as $X^*;(R^* \rightarrow \mathsf{f}) \vdash \mathsf{f}$, we immediately obtain the assertion for $(\curlywedge_\mathrm{c})$. For $(\Rightarrow)$ and $(\Rrightarrow)$ we must show that

$$\frac{X^*, Y^* \vdash R^*}{\overline{X^* \vdash Y^* \supset R^*}} \qquad \text{and} \qquad \frac{X^*; Y^* \vdash R^*}{\overline{X^* \vdash Y^* \rightarrow R^*}}$$

are admissible in RS, which follows by using the inference schemata for ⊃ and →. It remains to show that for each primitive rule R of $\mathrm{SCR_c}$ the sequent $\emptyset \vdash R^*$ can be derived in RS. We consider as examples →-introduction and ∨-elimination. For →-introduction we must derive in RS:

$$\emptyset \vdash (A \rightarrow B) \supset (A \rightarrow B) \ ,$$

which is trivial. For ∨-elimination we must derive in RS:

$$\emptyset \vdash ((A \vee B) \,\&(A \supset C) \,\&(B \supset C)) \supset C \ ,$$

which can easily be achieved by using &-, ⊃-, ∨-elimination and ⊃-introduction. □

§ 5. S4-modality

In systems of natural deduction, necessity (□) in the sense of S4 is characterized by an introduction rule, which allows one to pass over from A to $\Box A$, provided all assumptions on which A depends are modalized (see e.g. Prawitz 1965, Ch. VI). In its simplest form, “modalized” means that all assumptions start with □. Obviously, what one is using here is a new type of rule which is different from those formulated with the help of the double arrow and the triple arrow. To incorporate it into our structural framework, we must enlarge our definition of a rule and add structural inference schemata characterizing this new type of a rule. These inference schemata must reflect somehow the idea of modalization of assumptions.

We start with the sequent calculus for relevance logic $\mathrm{SCR_c}$ and add to the definition of rules and bunches the following clause:

If R is a rule, then so is $\rhd R$.

The operator "$\rhd$" is to express necessity on the structural level. Its indended meaning is that to establish a rule of the form $\rhd R$ under certain assumptions it is sufficient to establish R, provided all assumptions are modalized. Since assumptions have the structure of a bunch, we must define what it means for a bunch to be modalized:
Every rule of the form $\rhd R$ is modalized.

The empty bunches $\emptyset$ and $\curlyvee$ are modalized.
If X and Y are modalized, then so are (X, Y) and $(X; Y)$.

Note that the absurdity bunches $\curlywedge$ and (if present) # are *not* called modalized, nor are the rules $X \Rrightarrow R$ or $X \Rightarrow R$ for modalized X and R. This situation would be different if we wanted to reconstruct S5 rather than S4.

The characteristic inference schemata for $\rhd$ are the following:

$$\frac{X \vdash R}{X \vdash \rhd R}\,(\rhd+) \quad \text{provided } X \text{ is modalized}$$

$$\frac{X \vdash \rhd R}{X \vdash R}\,(\rhd-)\ .$$

On the logical level, we can characterize the logical constant $\Box$ by the following rule schemata:

$$\rhd p \Rightarrow \Box p \qquad (\Box\text{-I})$$

$$(\Box p, (\rhd p \Rightarrow r)) \Rightarrow r,\ \text{ equivalently: } \Box p \Rightarrow \rhd p\ . \qquad (\Box\text{-E})$$

The resulting system corresponds to $R^{\Box}$, which is nowadays considered to combine features of relevance with those of necessity in a more appropriate way than Anderson & Belnap's (1975) system E of entailment (see Read 1988). The functional completeness of the standard connectives, which now include $\Box$, is proved as in Ch. 5 § 2 with $\Box$ expressing $\rhd$.

If we restrict ourselves to the non-relevant fragment of this framework (i.e., disregarding the triple arrow, the semicolon, $\curlyvee$ and $\curlywedge$, and using in addition $\bot$ as a constant with the empty set of assertibility conditions and $\bot \Rightarrow r$ as elimination rule), we obtain intuitionistic S4.[7]

[7]Similar to Došen's (1980, 1985) idea of sequents of higher levels, one could consider a sequence $\rhd_0, \rhd_1, \rhd_2, \ldots$ of structural modal operators. For each natural number n and any rule R, $\rhd_n R$ would then

be defined as a rule, whereby $\rhd_0$ is identified with R. We could then define *n-modalized bunches* for every natural number n as follows:

A rule $\rhd_m R$ for $m \geq n$ is n-modalized.
$\emptyset$ and $\curlyvee$ are n-modalized for any n.
If X and Y are n-modalized, then so are (X, Y) and $(X; Y)$.
(In particular, if X is n-modalized, it is m-modalized for every $m \geq n$.)

As inference schemata governing $\rhd_n$ we could postulate:

$$\frac{X \vdash \rhd_n R}{X \vdash \rhd_{n+1} R}\,(\rhd_n +) \qquad \text{provided } X \text{ is } (n+1)\text{-modalized}$$

$$\frac{X \vdash \rhd_{n+1} R}{X \vdash \rhd_n R}\,(\rhd_n -)$$

This would lead to an infinity of modal constants of S4 type in the object language. (Here $\rhd_1$ with the inference schemata $(\rhd_0 +)$ and $(\rhd_0 -)$ would correspond to $\rhd$ with the inference schemata $(\rhd+)$ and $(\rhd-)$.)

Chapter 6

Judgements of Higher Levels in Martin-Löf's Logical Theory

Martin-Löf's intutionistic type theory is an approach which is being widely discussed in proof theory and (even more so) in theoretical computer science. As a framework for constructive mathematics it is a far-reaching theory. Via the interpretation of propositions as types of their proofs (an idea which originated in Howard 1980), it also contains a theory of pure logic, which deviates from standard treatments of intuitionistic logic in certain respects. Its central feature is the introduction of "A is a proposition" as a new form of judgement in addition to "A is true". This corresponds ot "A is a type" in addition to "a is an element of type A" in type theory.[1] The logical theory can be developed independently of the propositions-as-types approach, as carried out in Martin-Löf (1983, 1985a) and Smith (1984).

When the structural features of Martin-Löf's logical system are spelled out, it turns out that they fit very well into the range of ideas we are dealing with here. More precisely, underlying Martin-Löf's logical theory is a certain kind of structural framework that governs the reasoning with "judgements" in his sense and that can easily be extended to include "judgements of higher levels". These judgements of higher levels correspond to what we have before called "rules of higher levels".[2] The most characteristic feature of this structural framework is its "presuppositional" nature: In many cases assumptions can only be introduced provided certain other judgements have been proved. By use of higher-level judgements general schemata for "primitive judgements" (also called "axioms") for n-ary logical constants can be given which generalize (and standardize) the logical axioms or inference schemata given by Martin-Löf. They correspond to the schemata for primitive rules for logical constants of Ch. 2.

We concentrate on propositional logic, since the main philosophical ideas that distinguish Martin-Löf's framework from that presented in Ch. 1 already appear in the propositional case. This concentration also enables us to put aside the foundational questions of type theory which would come in if we had to deal with certain types as domains of

[1]For a presentation of type theory see Martin-Löf (1975, 1982, 1984) and Beeson (1985, Ch. XI).

[2]Judgements of second level are used in Martin-Löf (1980).

quantifiers. Furthermore, we do not combine the features of Martin-Löf's system with those of relevance logic, although this would be possible in principle, i.e. we present the system as a conceptual refinement of the (extensional) framework developed in Ch. 1. As a starting point, which is to be modified, we use the formalism SC2+, which is a system with rules as conclusions of rules. For the development of Martin-Löf's system it is not crucial to work in such a framework (as it was in the case of relevance logic). However, since we shall quite often use expressions like $X \Rightarrow R$, it is more convenient to treat them as they stand and not as abbreviations.

When we speak of Martin-Löf's system, we do not mean any particular formalism Martin-Löf has given himself. Rather, we mean the idea of a certain system that we are trying to reconstruct.

§ 1. Judgements and their presuppositions

As remarked in the introduction, Martin-Löf's logic and type theory belong to an approach to proof theory which considers formal proofs as representations of arguments by which we acquire and communicate knowledge. Like our preceding investigations, such an approach often tacitly assumes that formulae stand for propositions in a straightforward sense, so that a derivation of a formula can be interpreted as a demonstration of the truth of a corresponding proposition. Martin-Löf's central claim is that this view is too simple-minded, since it takes for granted that "proposition" is a meaningful concept independent of whether a system for formal reasoning has been constructed.

According to Martin-Löf, not everything that looks like a proposition is in fact a proposition, and whether it is a proposition may still depend on whether some other proposition has been established as true. This is to say,

(i) whether something is a proposition is itself a matter of argument, and

(ii) arguments for something's being a proposition cannot always be kept apart from arguments concerning truth.

Thesis (i) seems to be rather uncontroversial—there may obviously be discussions about whether something makes sense as a proposition or not. Thesis (ii), however, is strongly challenging. If arguments concerning being a proposition and being true are intertwined, then the usual structure of argumentation is affected. The entities we are arguing for in one and the same context are no longer of one kind, but of two kinds. Martin-Löf calls them "forms of judgements" A prop ("A is a proposition") and A true ("A is true"). Here "A prop" and "A true" are not metalinguistic assertions *about* A. Rather, the qualifications "true" and "prop" denote different ways of forming assertible contents.

Since "A is a proposition" means that it *makes sense* to ask whether A is true, this implies that *questions concerning sense and meaningfulness* and *questions concerning truth* are to be discussed on the *same level.* If we argue that A is a proposition (which is a presupposition for A's being true), we remain within the same context in which we try to establish the truth of A. Instead of having two levels of argumentation ("Sinnebene" vs.

"Geltungsebene" in traditional German terminology), we have one level with two forms of judgements.

One consequence of dealing with questions of truth and questions of meaningfulness in the same framework is the following: In Martin-Löf's fully developed system including type theory, it is possible to formally conclude from the well-known paradoxes (like Russell's) that certain assertions are not meaningful, i.e. that certain expressions are not propositions. These conclusions are results *within* the system. Therefore, the paradoxes are avoided in a natural way, without the need for restricting the system from outside by purely syntactic criteria, as done e.g. in ramified type theory.

When trying to codify Martin-Löf's philosophical approach in a formal framework we proceed as follows. We can no longer start from a definite range of formulae (usually inductively defined by a set of formations rules) and then give inference schemata or rules for the derivations of formulae (or rules of higher levels built up from formulae). Rather, we must start with closed expressions A and introduce a system of inference schemata for judgements of the form A prop and A true (and more complex judgements). Of course, these closed expressions may formally be called "formulae". However, since they cannot, as in usual presentation of logical formalisms, be intuitively conceived as standing for propositions, it is more appropriate to keep "closed expression" as the standard term. (Whether an expression is closed or not is a matter of syntax or, as Martin-Löf would say, of a "theory of expressions".) The judgements "A prop" and "A true", being composed of a closed expressions and an indicator of a form of judgement, are not themselves considered closed expressions, so that an iteration like "A prop prop" is not allowed.

Formally, judgements of higher levels and sequents are defined as follows. Suppose a basic range of closed expressions is given, metalinguistically denoted by A, B, C, D (with and without indices). Then A prop and A true are judgements (also called "*categorical judgements*"). If X is a finite sequence of judgements and R is a judgement, then $(X \Rightarrow R)$ is a judgement (also called "*hypothetical judgement*"). R is called the conclusion and the elements of X are called the premises of $(X \Rightarrow R)$. Judgements are identified with one-element sequences of judgements. A sequent is of the form $X \vdash R$ with a finite sequence of judgements X as the antecedent and a judgement R as the succedent. Contrary to Ch. 1, where we had sets to the left of the rule arrow and of the turnstile, and to Ch. 5, where we had bunches built up by binary operations, we here work with finite sequences, since the order of their elements will be important.

In the following, we use R (sometimes primed and with indices) as syntactic variables for judgements, and X, Y, Z (with and without indices) as syntactic variables for finite sequences of judgements. By $\langle\rangle$ we denote the empty sequence of judgements (which is considered a limiting case of a finite sequence). Outer brackets in the notation of hypothetical judgements can be omitted. We use $X \Rightarrow Y$ as an abbreviation for the sequence $(X \Rightarrow R_1), \ldots, (X \Rightarrow R_n)$ and $X \vdash Y$ for $X \vdash R_1, \ldots, X \vdash R_n$, if Y is $R_1, \ldots, R_n$. $X \Rightarrow \langle\rangle$ and $X \vdash \langle\rangle$ stand for the empty sequence $\langle\rangle$. Throughout we use the comma both to divide the elements of a sequence and to denote the concatenation of sequences. For example,

$$X, Y, R, Z, R'$$

denotes the sequence

$$R_1, \dots, R_n, R'_1, \dots, R'_m, R, R''_1 \dots, R''_k, R',$$

if X denotes the sequence $R_1, \dots, R_n, Y$ the sequence $R'_1, \dots, R'_m$ and Z the sequence $R''_1, \dots, R''_k$.

With each judgement R and each finite sequence of judgements X we associate a finite sequence of judgements as its *presupposition*, denoted by $\mathbb{P}(R)$ and $\mathbb{P}(X)$, respectively. Intuitively, $\mathbb{P}(R)$ or $\mathbb{P}(X)$ is to be understood as what has to be proved for R to be a meaningful judgement or for X to be a meaningful sequence of judgements. (*Formally*, "meaningfulness" will be *defined* via the $\mathbb{P}$-operation.) The definition of $\mathbb{P}$ reflects the idea that presuppositions concerning meaningfulness are to be treated within our structural system itself. According to what was said before, it should be clear that $\mathbb{P}(A \ \mathsf{prop}) = \langle\rangle$ and $\mathbb{P}(A \ \mathsf{true}) = A \ \mathsf{prop}$. It makes always sense to assert $A \ \mathsf{prop}$ (even if this assertion turns out incorrect), i.e., $A \ \mathsf{prop}$ has no presupposition. But one may discuss the correctness of $A \ \mathsf{true}$ only if A is a proposition, so $A \ \mathsf{prop}$ is a presupposition for $A \ \mathsf{true}$.

The simplest way of defining the presuppositions for hypothetical judgements or sequences of judgements would be to consider them to be meaningful if all the categorical judgements occurring in them are meaningful, i.e. to define: $\mathbb{P}(\langle\rangle) = \langle\rangle$, $\mathbb{P}(R_1, \dots, R_n) = \mathbb{P}(R_1), \dots, \mathbb{P}(R_n)$, $\mathbb{P}(X \Rightarrow R) = \mathbb{P}(X), \mathbb{P}(R)$. However, by such a definition we would immediately give up much expressive power. According to Martin-Löf, the meaningfulness of the elements of a sequence of judgements need not be established independently for each element, but may be established step by step. (Here it comes in that we work with finite sequences rather than sets of judgements.) This "sequential" procedure is different from the "parallel" one since now the presuppositions of one element of a sequence may depend on earlier elements.

To make this intuitively clear, consider a hypothetical judgement $R_1, \dots, R_n \Rightarrow R$. Its *intended* meaning is that it allows one to establish R provided $R_1, \dots, R_n$ have been established. Therefore it should be considered meaningful if it leads from meaningful judgements to meaningful judgements. Thus for $R_1, \dots, R_n \Rightarrow R$ to be meaningful it would be too strong to require that R be meaningful outright. The question of meaningfulness of R does not arise *before* we have already established $R_1, \dots, R_n$. So R need only be meaningful *provided* the judgements $R_1, \dots, R_n$ have been established. Furthermore, the judgements $R_1, \dots, R_n$ may be established step by step, i.e. first R_1, and then R_2, etc. This means that for the sequence $R_1, \dots, R_n$ the question of meaningfulness of R_2 does not arise before R_1 has been established, that of R_3 not before R_1 and R_2 have been established, and so on.

This motivation leads to the following definition of the presuppositions of a judgement or of a sequence of judgements:

$\mathbb{P}(A \ \mathsf{prop}) = \langle\rangle$
$\mathbb{P}(A \ \mathsf{true}) = A \ \mathsf{prop}$
$\mathbb{P}(\langle\rangle) = \langle\rangle$
$\mathbb{P}(X \Rightarrow R) = \mathbb{P}(X), (X \Rightarrow \mathbb{P}(R))$

$\mathbb{P}(X, R) = \mathbb{P}(X), (X \Rightarrow \mathbb{P}(R))$.

For example,

$$\mathbb{P}((A\ \mathsf{true}, B\ \mathsf{true}) \Rightarrow C\ \mathsf{true}) = A\ \mathsf{prop}, (A\ \mathsf{true} \Rightarrow B\ \mathsf{prop}), (A\ \mathsf{true}, B\ \mathsf{true}) \Rightarrow C\ \mathsf{prop}.$$

A judgement R or a sequence of judgements X is called *meaningful*, if its presupposition can be derived, i.e. if $\langle\rangle \vdash R$ or $\langle\rangle \vdash X$, respectively, can be derived in the system being developed.

We have used the term "judgement" to denote something that need not be meaningful. From the philosophical point of view it would be more appropriate to use a term like "candidate of judgements" for what we have now called "judgement" and to reserve the term "judgement" for something that is meaningful. The motivation for our choice of terminology was only to have a short word for what has the syntactic form of a judgement.

Furthermore, we have not presented here Martin-Löf's philosophical ideas about the precise sense in which A prop is a presupposition for A true. These ideas are based on principles concerning verification and knowledge which are connected to an epistemology of transcendental idealism (see Martin-Löf 1985a,b). To a great extent our system can be intuitively motivated and formally developed by just relying on the fact *that* judgements R have other judgements or sequences $\mathbb{P}(R)$ thereof as their presuppositions. This may make Martin-Löf's logical system accessible to people who do not share his epistemological claims.[3]

§ 2. The structural framework with judgements of higher levels

Our structural inference schemata result from that of SC2+ by taking the specific features of Martin-Löf's approach into account. These are in particular (i) that antecedents of sequents and premises of hypothetical judgements are sequences rather than sets, and (ii) that judgements or sequences of judgements are meaningful only if their presuppositions have been established. Due to (i), a schema of contraction is added. A schema of permutation, which one would also expect in this context, is admissible. Due to (ii), at several places additional upper sequents must be required in the inference schemata to ensure that new judgements introduced into a derivation are meaningful. We assume a set of primitive judgements to be given. These primitive judgements, which correspond to primitive rules in the previous chapters, are also called "axioms". How primitive judgements for logical constants look like, is discussed in the next section.

The system SCML ("sequent calculus for Martin-Löf's logical theory") has the following inference schemata:

$$\frac{X, R, Y, R, Z \vdash R'}{X, R, Y, Z \vdash R'}\ (\text{Contr}) \qquad\qquad \frac{X, Y \vdash R \quad X \vdash \mathbb{P}(Z)}{X, Z, Y \vdash R}\ (\text{Thin})$$

[3]It is planned to deal with these semantical and epistemological questions at another place.

$$\frac{X \vdash \mathbb{P}(R)}{X, R \vdash R}\,(\text{Refl}) \qquad\qquad R\,\frac{X \vdash \mathbb{P}(R)}{X \vdash R}\,(\text{Prim}) \qquad (R \text{ Axiom})$$

proviso for applications of (Refl) and (Prim): X must be $\langle\rangle$ if $\mathbb{P}(R)$ is $\langle\rangle$

$$\frac{X \vdash Y \quad X \vdash Y \Rightarrow R}{X \vdash R}\,(\text{App}) \qquad\qquad \frac{X, Y \vdash R}{X \vdash Y \Rightarrow R}\,(\Rightarrow +)\ .$$

Since $X \vdash \mathbb{P}(Z)$ and $X \vdash \mathbb{P}(R)$ are sequences of sequents rather than of sets, the upper sequents of an inference should now be considered to form a sequence. Obviously, this is no essential difference to the procedure in Ch. 1, since derivations of upper sequents can always be re-ordered. Correspondingly, when we say that a sequence of sequents is derivable in SCML, this means the same as saying that every element of this sequent is derivable. The empty sequence $\langle\rangle$ of sequents is always considered derivable.

We motivate the schemata of SCML in relation to those of SC2+ by using a meaningfulness property for sequents. A sequent $X \vdash R$ is called *meaningful* if $\langle\rangle \vdash \mathbb{P}(X, R)$ is derivable in SCML. We want to obtain as a result that only meaningful sequents are derivable (see below Theorem 6.2).

(Contr) We only allow for contraction to the left and not to the right, i.e., we do not have

$$\frac{X, R, Y, R, Z \vdash R'}{X, Y, R, Z \vdash R'}\,(\text{Contr}_\text{r})\ .$$

The upper sequent of this schema may be meaningful only because for some element R_1 of Y, the sequent $X, R, Y_1 \vdash \mathbb{P}(R_1)$ is derivable, where Y is Y_1, R_1, Y_2, i.e., $\mathbb{P}(R_1)$ may depend on R as one of the previous elements of the antecedent. If $X, Y_1 \vdash \mathbb{P}(R_1)$ is not derivable, this meaningfulness is not carried over to the lower sequent.

In view of $(\Rightarrow +)$ and its converse, which is admissible, the Z in the schema (Contr) can be omitted. In sequent calculi with sequences as antecedents one usually has a permutation schema like

$$\frac{X, R_1, R_2, Y \vdash R}{X, R_2, R_1, Y \vdash R}$$

at one's disposal. This schema is not admissible in SCML, for reasons analogous to those speaking against (Contr$_\text{r}$): The meaningfulness of the upper sequent requires the derivability of $X, R_1 \vdash \mathbb{P}(R_2)$, i.e., $\mathbb{P}(R_2)$ may depend on R_1, and if $X \vdash \mathbb{P}(R_2)$ is not derivable, this meaningfulness is not carried over to the lower sequent. However, by adding $X \vdash \mathbb{P}(R_2)$ as an upper sequent, we obtain the schema

$$\frac{X, R_1, R_2, Y \vdash R \quad X \vdash \mathbb{P}(R_2)}{X, R_2, R_1, Y \vdash R}\,(\text{Perm})\ ,$$

which can be shown to be admissible in SCML: From its upper sequents we obtain $X, R_2, R_1, R_2, Y \vdash R$ by (Thin), and from that $X, R_2, R_1, Y \vdash R$ by (Contr).

(Thin) If a sequence of judgements Z is to be inserted into the antecedent of a sequent, it

must be guaranteed that meaningfulness is not violated. Therefore it is required by means of the right upper sequent that the presuppositions of Z be fulfilled, provided those judgements which stand left to the place where Z is to be inserted are established. It can be shown that (Thin) is equivalent to the schema which has only a single judgement R' at the place of the sequence of judgements Z.
(Refl) The judgement R can only be introduced as an assumption if it makes sense, i.e. if its presuppositions have been established. Since these presuppositions may themselves have been established under assumptions, we permit assumptions X in the antecedents, i.e., we do not only consider the special case

$$\frac{\langle\rangle \vdash \mathbb{P}(R)}{R \vdash R}$$

of (Refl). It is the purpose of the proviso to avoid something like

$$\frac{}{X, A \text{ prop} \vdash A \text{ prop}}$$

as a special cases of the inference schema, no matter whether X is meaningful or not. This problem arises from the fact that $X \vdash \langle\rangle$ is defined to be $\langle\rangle$.
(Prim) Here analogous comments as for (Refl) apply.
(App) and ($\Rightarrow +$) These schemata exactly correspond to what we had in SC2+.

Lemma 6.1 For all X and Y:

(i) $\mathbb{P}(X, Y) \dashv\vdash \mathbb{P}(X), (X \Rightarrow \mathbb{P}(Y))$ is derivable in SCML.

(ii) If $X \vdash \mathbb{P}(Y)$ is derivable in SCML, then so is $X, Y \vdash Y$.

Proof

(i) We proceed by induction on the length of the sequence Y. If Y is $\langle\rangle$, nothing is to prove. If Y is a judgement R, the assertion holds by definition. If Y is of the form Z, R, then

$$\mathbb{P}(X, Y) = \mathbb{P}(X, Z), ((X, Z) \Rightarrow \mathbb{P}(R)) \ ,$$

and

$$X \Rightarrow \mathbb{P}(Y) \dashv\vdash X \Rightarrow \mathbb{P}(Z), ((X, Z) \Rightarrow \mathbb{P}(R))$$

is derivable. By induction hypothesis we have that

$$\mathbb{P}(X, Z) \dashv\vdash \mathbb{P}(X), (X \rightarrow \mathbb{P}(Z))$$

is derivable, which immediately yields the assertion.

(ii) We use again induction on the length of Y. If Y is $\langle\rangle$, nothing is to prove. If Y is a judgement R, then the assertion follows by (Refl). If Y is of the form Z, R, then from $X \vdash \mathbb{P}(Z, R)$ we obtain $X \vdash \mathbb{P}(Z)$ and $X, Z \vdash \mathbb{P}(R)$, hence by induction hypothesis $X, Z \vdash Z$, by (Thin) $X, Z, R \vdash Z$, and by (Refl) $X, Z, R \vdash R$. Thus we have derived $X, Y \vdash Y$. □

The following Theorem formally establishes what in the comments on the structural schemata was referred to as a desirable feature of our system. We write $\mathbb{P}\mathbb{P}(X)$ as an abbreviation for $\mathbb{P}(\mathbb{P}(X))$.

Theorem 6.2 (i) For all X: $\langle\rangle \vdash \mathbb{P}\mathbb{P}(X)$ is derivable in SCML.

(ii) If $X \vdash R$ is derivable in SCML, then so is $\langle\rangle \vdash \mathbb{P}(X, R)$.

Proof

(i) If X is $\langle\rangle$, the assertion is trivial. If X is a categorical judgement, then $\mathbb{P}\mathbb{P}(X) = \langle\rangle$ by definition. If X is a judgement of the form $Y \Rightarrow R$, then $\mathbb{P}\mathbb{P}(X \Rightarrow R) = \mathbb{P}\mathbb{P}(X, R)$. Thus it suffices to show that $\langle\rangle \vdash \mathbb{P}\mathbb{P}(Y, R)$ is derivable, provided $\langle\rangle \vdash \mathbb{P}\mathbb{P}(Y)$ and $\langle\rangle \vdash \mathbb{P}\mathbb{P}(R)$ are derivable. Now we have

$$\begin{aligned}\mathbb{P}\mathbb{P}(Y, R) &= \mathbb{P}(\mathbb{P}(Y), (Y \Rightarrow \mathbb{P}(R))) \\ &= \mathbb{P}\mathbb{P}(Y), (\mathbb{P}(Y) \Rightarrow \mathbb{P}(Y \Rightarrow \mathbb{P}(R))) \\ &= \mathbb{P}\mathbb{P}(Y), (\mathbb{P}(Y) \Rightarrow (\mathbb{P}(Y), (Y \Rightarrow \mathbb{P}\mathbb{P}(R)))) \ .\end{aligned}$$

By hypothesis $\langle\rangle \vdash \mathbb{P}\mathbb{P}(Y)$ is derivable. Therefore $\mathbb{P}(Y) \vdash \mathbb{P}(Y)$ is derivable by Lemma 6.1 (ii). Since $\langle\rangle \vdash \mathbb{P}\mathbb{P}(R)$ is derivable by hypothesis, we obtain $\mathbb{P}(Y) \vdash \mathbb{P}\mathbb{P}(R)$ by (Thin) (using $\langle\rangle \vdash \mathbb{P}\mathbb{P}(Y)$) and $\mathbb{P}(Y), Y \vdash \mathbb{P}\mathbb{P}(R)$ again by (Thin) (using $\mathbb{P}(Y) \vdash \mathbb{P}(Y)$).

(ii) We proceed by induction on the length of derivations. If this length is 1, then (Refl) or (Prim) are used without upper sequents. Then $\mathbb{P}(R) = \langle\rangle$, and $\langle\rangle \vdash \mathbb{P}(R, R)$ and $\langle\rangle \vdash \mathbb{P}(R)$ are empty sequences. If this length is greater than 1, we distinguish cases according to the inference used in the last step, supposing the assertion holds for the upper sequents of this inference. We use the same syntactic variables as in the presentation of the inference schemata above.
(Contr) If $\langle\rangle \vdash \mathbb{P}(X, R, Y, R, Z, R')$ is derivable, then by Lemma 6.1 (i) so is $\langle\rangle \vdash \mathbb{P}(X, R, Y)$ and $X, R, Y \vdash \mathbb{P}(R, Z, R')$. From the latter sequent we obtain $X, R, Y, R \vdash \mathbb{P}(Z, R')$, from which $X, R, Y \vdash \mathbb{P}(Z, R')$ follows by (Contr). Together with $\langle\rangle \vdash \mathbb{P}(X, R, Y)$ this means $\langle\rangle \vdash \mathbb{P}(X, R, Y, Z, R')$.
(Thin) If $\langle\rangle \vdash \mathbb{P}(X, Y, R)$ is derivable, then by Lemma 6.1 (i) so is $\langle\rangle \vdash \mathbb{P}(X)$ and $X \vdash \mathbb{P}(Y, R)$. From the latter sequent, using $X \vdash \mathbb{P}(Z)$, we obtain by (Thin) $X, Z \vdash \mathbb{P}(Y, R)$. Since $X \vdash \mathbb{P}(Z)$ and $\langle\rangle \vdash \mathbb{P}(X)$ yield $\langle\rangle \vdash \mathbb{P}(X, Z)$, we obtain $\langle\rangle \vdash \mathbb{P}(X, Z, Y, R)$.

(Prim) and (Refl) From $\langle\rangle \vdash \mathbb{P}(X, \mathbb{P}(R))$ we obtain $\langle\rangle \vdash \mathbb{P}(X)$ and thus, together with $X \vdash \mathbb{P}(R)$, $\langle\rangle \vdash \mathbb{P}(X, R)$. From $X \vdash \mathbb{P}(R)$ we obtain by (Thin) also $X, R \vdash \mathbb{P}(R)$, so that we have $\langle\rangle \vdash \mathbb{P}(X, R, R)$.
(App) From $\langle\rangle \vdash \mathbb{P}(X, Y, R)$ we obtain by Lemma 6.1 (i) $\langle\rangle \vdash \mathbb{P}(X)$ and $X \vdash \mathbb{P}(Y, R)$, thus $X \vdash Y \Rightarrow \mathbb{P}(R)$. Together with $X \Rightarrow Y$, (App) yields $X \vdash \mathbb{P}(R)$. Hence we have derived $\langle\rangle \vdash \mathbb{P}(X, R)$.
($\Rightarrow$+) trivial. □

Part (i) of this theorem implies that a sequent which states that something is meaningful is itself always meaningful. Part (ii) says that in SCML only meaningful sequents are derivable. Note that this theorem holds with respect to *any* set of primitive judgements, i.e., it expresses a property of the structural apparatus.

§ 3. Axioms for logical constants

Our procedure is similar to that of Ch. 2. We consider the closed expressions A occurring in categorical judgements A prop and A true to be built up from atomic closed expressions by use of n-ary logical constants S which are given as a sequence

$$S_1, S_2, \ldots, S_i, \ldots$$

Let $\underline{p}$ be $p_1, \ldots, p_n$. By an n-ary *judgement schema* ϱ (or $\varrho[\underline{p}]$) we understand something that is built up like a judgement, but only contains the schematic letters $p_1, \ldots, p_n$ instead of atomic closed expressions. For example,

$$p_1 \ \mathsf{true} \Rightarrow S(p_1, p_2) \ \mathsf{prop}$$

is a binary judgement schema, if S is a binary logical constant. An *assertibility condition* for an n-ary logical constant S is a finite sequence $\Delta[\underline{p}]$ of n-ary judgement schemata, satisfying the following requirement:

($*$) All judgement schemata in $\Delta[\underline{p}]$ are built up from judgement schemata of the form "α true", i.e., "... prop" does not occur in $\Delta[\underline{p}]$.

The operation $\mathbb{P}$ is defined for judgement schemata and assertibility conditions in the same way as for judgements and sequences of judgements. With each S in the sequence of constants a finite set

$$\mathbb{D}(S) = \{\Delta_1[\underline{p}], \ldots, \Delta_m[\underline{p}]\}$$

of assertibility conditions is assumed to be associated. If S is S_i, only logical constants S_j for $j < i$ are allowed to occur in $\mathbb{D}(S)$. As in Ch. 2 (but unlike the situation in relevance logic, see Ch. 5), we allow for $\mathbb{D}(S)$ to be empty.

Condition ($*$) is crucial. If we allowed for "... prop" to occur in assertibility conditions for logical constants, we would be able to construct a system-internal definition of the category of propositions, i.e. we could define a unary constant P such that for each closed expression A we can derive A $\mathsf{prop} \dashv\vdash P(A)$ true. From that we would obtain a system-internal definition of the category of true propositions, i.e., we could define a unary constant T such that for all closed expressions A the sequents $(A\ \mathsf{prop}, A\ \mathsf{true}) \vdash T(A)\ \mathsf{true}$ and $(A\ \mathsf{prop}, T(A)\ \mathsf{true}) \vdash A\ \mathsf{true}$ can be derived (we simply have to take $T(A)$ to be $P(A)\ \&\ A$). Such an internal truth-definition would make Martin-Löf's system, when enlarged with type theory, inconsistent, as shown by Martin-Löf (1983)[4] and Aczel (1980)[5].

Suppose $\mathbb{D}(S) = \{\Delta_1[\underline{p}], \ldots, \Delta_m[\underline{p}]\}$. Then we define $\mathbb{P}(S)$ to be the sequence $\mathbb{P}(\Delta_1[\underline{p}]), \ldots, \mathbb{P}(\Delta_m[\underline{p}])$, based on an arbitrary (but fixed) order of the elements of $\mathbb{P}(S)$ (Theorem 6.3 below will show that it is irrelevant which specific order is chosen.) $\mathbb{P}(S)$ is defined to be $\langle\rangle$ if $\mathbb{D}(S)$ is empty. If $\underline{A}$ is $A_1, \ldots, A_n$, let $\mathbb{P}(S)[\underline{A}]$ be $\mathbb{P}(\Delta_1[\underline{A}])$, $\ldots, \mathbb{P}(\Delta_m[\underline{A}])$. In Martin-Löf's system the axioms for logical constants contain in addition to introduction and elimination axioms *formation axioms*, which guarantee that the conclusions of the introduction axioms are meaningful, provided their premises are meaningful. The schema of an S-formation axiom is the following:

$$\mathbb{P}(S) \Rightarrow S(\underline{p})\ \mathsf{prop} \qquad (S\text{-F})$$

The premises of the S-formation axioms pertain to the S-introduction and S-elimination axioms. Their schemata are the following:

$$\begin{array}{l} \mathbb{P}(S), \Delta_1[\underline{p}] \Rightarrow S(\underline{p})\ \mathsf{true} \\ \qquad\vdots \\ \mathbb{P}(S), \Delta_m[\underline{p}] \Rightarrow S(\underline{p})\ \mathsf{true} \end{array} \qquad (S\text{-I})$$

$$\begin{array}{r} (\mathbb{P}(S), (S(\underline{p})\ \mathsf{true} \Rightarrow \mathbb{P}(r)), S(\underline{p})\ \mathsf{true}, (\Delta_1[\underline{p}] \Rightarrow r), \\ \ldots, (\Delta_m[\underline{p}] \Rightarrow r)) \Rightarrow r\ . \end{array} \qquad (S\text{-E})$$

Here r is a schematic letter for arbitrary judgements. If r is instantiated by the judgement R, then $\mathbb{P}(r)$ is to be replaced by $\mathbb{P}(R)$. The following theorem shows that all formation, introduction and elimination axioms are meaningful. Its proof shows how this result is achieved by the additional premises $\mathbb{P}(S)$ of (S-I) and (S-E) and the premise $S(\underline{p})\ \mathsf{true} \Rightarrow \mathbb{P}(r)$ of (S-E) together with (S-F). Furthermore, the order of the elements of $\mathbb{P}(S)$ is not important nor is the order of the premises of the form $\Delta_i[\underline{p}] \Rightarrow r$ in (S-E). Therefore, also in the context of Martin-Löf's logic it is justified to consider $\mathbb{D}(S)$ to be a *set* of assertibility conditions.

[4]Unfortunately, Martin-Löf does not explicitly point to this problem when introducing logical constants and dealing with the form of truth conditions of logically complex expressions, doing so only at a later stage. In the context of type theory this problem does not arise, since the forms of judgement corresponding to A prop and A true, viz. A type and $a \in A$, respectively, are syntactically constructed in a different way. Since unlike $a \in A$, the judgement A type does not carry a construction term with itself, it cannot be used in the premises of introduction rules of type-theoretical operators.

[5]In particular, Aczel could show that it is just this possibility of an internal truth-definition which makes the system of Frege's "Grundgesetze der Arithmetik" (1893) inconsistent when this system is reconstructed in a similar setting.

Theorem 6.3 (i) If R is an instance of (S-F), (S-I) or (S-E), then $\langle\rangle \vdash \mathbb{P}(R)$ is derivable in SCML.

(ii) Let R be an instance of (S-F), (S-I) or (S-E). Let R' result from R by permuting elements of $\mathbb{P}(S)[\underline{A}]$ within the premise of R, and, in the case of (S-E), of permuting premises of the form $\Delta_i(\underline{A}) \Rightarrow R_1$ of R. Then $R \dashv\vdash R'$ is derivable in SCML.

Proof

(i) Let R be an S-formation axiom $\mathbb{P}(S)[\underline{A}] \Rightarrow S(\underline{A})$ prop. Then $\mathbb{P}(R) = \mathbb{P}(\mathbb{P}(\Delta_1[\underline{A}]), \ldots, \mathbb{P}(\Delta_m[\underline{A}]))$. The assertion follows by use of Theorem 6.2 (i).
Let R be an S-introduction axiom $\mathbb{P}(S)[\underline{A}], \Delta_i[\underline{A}] \Rightarrow S(\underline{A})$ true. Then by Lemma 6.1 (i),

$$\mathbb{P}(R) = \mathbb{P}(\mathbb{P}(S)[\underline{A}]), (\mathbb{P}(S)[\underline{A}] \Rightarrow \mathbb{P}(\Delta_i[\underline{A}] \Rightarrow S(\underline{A}) \ \mathsf{true})).$$

Since $\langle\rangle \vdash \mathbb{P}(\mathbb{P}(S)[\underline{A}])$ and $\mathbb{P}(S)[\underline{A}] \vdash \mathbb{P}(\Delta_i[\underline{A}])$ are derivable, we must only show that $\mathbb{P}(S)[\underline{A}], \Delta_i[\underline{A}] \vdash S(\underline{A})$ prop is derivable, which follows by (S-F) and (Thin).
Let R be an S-elimination axiom

$$(\mathbb{P}(S)[\underline{A}], (S(\underline{A}) \ \mathsf{true} \Rightarrow \mathbb{P}(R')), S(\underline{A}) \ \mathsf{true}, (\Delta_1[\underline{A}] \Rightarrow R'), \ldots, (\Delta_m[\underline{A}] \Rightarrow R') \Rightarrow R'.$$

We know that $\langle\rangle \vdash \mathbb{P}(\mathbb{P}(S)[\underline{A}])$ as well as $\mathbb{P}(S)[\underline{A}] \vdash S(\underline{A})$ prop (by S-formation) and $\mathbb{P}(S)[\underline{A}] \vdash \mathbb{P}(\Delta_i[\underline{A}])$ (for all i, $1 \leq i \leq m$) are derivable, and since we have $S(\underline{A})$ $\mathsf{true} \Rightarrow \mathbb{P}(R')$ as a premise of R, we can derive $\mathbb{P}(S)[\underline{A}], S(\underline{A})$ $\mathsf{true} \vdash \mathbb{P}(\Delta_i[\underline{A}] \Rightarrow R')$. Hence we can conclude that $\langle\rangle \vdash R$ is derivable.

(ii) Since we can derive $\langle\rangle \vdash \mathbb{P}(\mathbb{P}(S)[\underline{A}])$, we can use the admissibility of (Perm) in SCML to conclude that the elements of $\mathbb{P}(S)[\underline{A}]$ can be exchanged in R. Concerning premises of the form $\Delta_i[\underline{A}] \Rightarrow R_1$ in S-elimination axioms, we use $\mathbb{P}(S)[\underline{A}]$, $S(\underline{A})$ $\mathsf{true} \vdash \mathbb{P}(\Delta_i[\underline{A}] \Rightarrow R_1)$ as obtained in (i). □

If our only axioms are formation, introduction and elimination axioms for logical constants, then according to this theorem the upper sequents of (Prim) can always be derived, so that we may always start a derivation with $\langle\rangle \vdash R$ for an axiom R.

The assertibility conditions and axioms for the standard connectives &, $\vee$, $\supset$, $\perp$ are presented in Table 7.

The logical system based on formation, introduction and elimination axioms for the standard connectives yields intuitionistic propositional logic in the following sense. For each expression $\alpha(p_1, \ldots, p_n)$ built up from the schematic letters $p_1, \ldots, p_n$ by means of the standard connectives the following holds: $\alpha(p_1, \ldots, p_n)$ is a valid formula of intuitionistic propositional logic iff p_1 $\mathsf{prop}, \ldots, p_n$ $\mathsf{prop} \vdash \alpha(p_1, \ldots, p_n)$ true is schematically derivable in SCML (i.e., derivable by using the schematic letters $p_1, \ldots, p_n$ as atomic expressions).

Functional completeness is similar to Ch. 2 § 3 demonstrated by using a translation g of assertibility conditions into logically compound expressions, which only use the standard

TABLE 7

Assertibility conditions and axioms for the standard propositional connectives in Martin-Löf's logical system

(The letters p and q stand for p_1 and p_2, respectively. The schematic letter r is to be instantiated by arbitrary judgements.)

$\mathbb{D}(\&) = \{(p\ \mathsf{true}, q\ \mathsf{true})\}$ $\qquad\mathbb{P} = p\ \mathsf{prop}, (p\ \mathsf{true} \Rightarrow q\ \mathsf{prop})$

$p\ \mathsf{prop}, (p\ \mathsf{true} \Rightarrow q\ \mathsf{prop}) \Rightarrow p \,\&\, q\ \mathsf{prop}$ (&-F)

$\mathbb{P}(\&), p\ \mathsf{true}, q\ \mathsf{true} \Rightarrow p \,\&\, q\ \mathsf{prop}$ (&-I)

$\mathbb{P}(\&), (p \,\&\, q\ \mathsf{true} \Rightarrow \mathbb{P}(r)), (p \,\&\, q\ \mathsf{true}), ((p\ \mathsf{true}, q\ \mathsf{true}) \Rightarrow r) \Rightarrow r$ (&-E)

equivalent: $\mathbb{P}(\&), p \,\&\, q\ \mathsf{true} \Rightarrow p\ \mathsf{true}$

$\mathbb{P}(\&), p \,\&\, q\ \mathsf{true} \Rightarrow q\ \mathsf{true}$

$\mathbb{D}(\vee) = \{p\ \mathsf{true}, q\ \mathsf{true}\}$ $\qquad\mathbb{P}(\vee) = p\ \mathsf{prop}, q\ \mathsf{prop}$

$p\ \mathsf{prop}, q\ \mathsf{prop} \Rightarrow p \vee q\ \mathsf{prop}$ (∨-F)

$\mathbb{P}(\vee), p\ \mathsf{true} \Rightarrow p \vee q\ \mathsf{true}$ (∨-I)

$\mathbb{P}(\vee), q\ \mathsf{true} \Rightarrow p \vee q\ \mathsf{true}$

$\mathbb{P}(\vee), (p \vee q\ \mathsf{true} \Rightarrow \mathbb{P}(r)), (p \vee q\ \mathsf{true}), (p\ \mathsf{true} \Rightarrow r), (q\ \mathsf{true} \Rightarrow r) \Rightarrow r$ (∨-E)

$\mathbb{D}(\supset) = \{p\ \mathsf{true} \Rightarrow q\ \mathsf{true}\}$ $\qquad\mathbb{P}(\supset) = p\ \mathsf{prop}, (p\ \mathsf{true} \Rightarrow q\ \mathsf{prop})$

$p\ \mathsf{prop}, (p\ \mathsf{true} \Rightarrow q\ \mathsf{prop}) \Rightarrow p \supset q\ \mathsf{prop}$ (⊃-F)

$\mathbb{P}(\supset), (p\ \mathsf{true} \Rightarrow q\ \mathsf{prop}) \Rightarrow p \supset q\ \mathsf{true}$ (⊃-I)

$\mathbb{P}(\supset), (p \supset q\ \mathsf{true} \Rightarrow \mathbb{P}(r)), (p \supset q\ \mathsf{true}), ((p\ \mathsf{true} \Rightarrow q\ \mathsf{true}) \Rightarrow r) \Rightarrow r$ (⊃-E)

equivalent: $\mathbb{P}(\supset), p \supset q\ \mathsf{true}, p\ \mathsf{true} \Rightarrow q\ \mathsf{true}$

$\mathbb{D}(\bot) = \emptyset$ $\qquad\mathbb{P}(\bot) = \langle\rangle$

$\langle\rangle \Rightarrow \bot\ \ \mathsf{prop}$ (⊥-F)

no ⊥-introduction axiom

$(\bot\ \mathsf{true} \Rightarrow \mathbb{P}(r)), \bot\ \mathsf{true} \Rightarrow r$ (⊥-E) connectives:

$g(\alpha\ \mathsf{true}) = \alpha$
$g(\langle\rangle) = \bot \supset \bot$
$g(\varrho_1, \ldots, \varrho_n) = g(\varrho_1) \,\&\, \ldots \,\&\, g(\varrho_n)$
$g(\Delta \Rightarrow \varrho) = g(\Delta) \supset g(\varrho)$.

It can easily be shown that $\mathbb{P}(\Delta) \vdash g(\Delta)\ \mathsf{prop}$ is schematically derivable[6] and that for any assertibility condition $\Delta[\underline{p}]$,

$(**)$ if $\langle\rangle \vdash \mathbb{P}(\Delta[\underline{A}])$ is derivable then so is $\mathbb{P}(\Delta[\underline{A}]) \dashv\vdash g(\mathbb{P}(\Delta[\underline{A}]))\ \mathsf{true}$.

Then we define

$h(S(\underline{p})) = g(\Delta_1[\underline{p}]) \vee \ldots \vee g(\Delta_m[\underline{p}])$ if $\mathbb{D}(S) \neq \emptyset$
$h(S(\underline{p})) = \bot$ if $\mathbb{D}(S) = \emptyset$.

Following the methods described in Ch. 2 § 3 it can be shown that for all $\underline{A}$, if $\langle\rangle \vdash \mathbb{P}(S)[\underline{A}]$ is derivable, then we can derive $S(\underline{A})\ \mathsf{true} \dashv\vdash h(S(\underline{A}))\ \mathsf{true}$ and hence replace S by an expression which only contains the standard connectives.[7]

An assertion like $(**)$ cannot be proved for arbitrary X instead of instances $\Delta[\underline{A}]$ assertibility conditions $\Delta[\underline{p}]$. If in such an X a judgement $A\ \mathsf{prop}$ occurs at some place, $g(X)$ (or a corresponding translation operating on sequences of judgements rather than judgement schemata) is not defined, since, as remarked above, we do not have a logical constant which expresses $A\ \mathsf{prop}$. The fact that in Martin-Löf's logical theory judgements of higher levels cannot always be translated into categorical judgements gives the concept of iterated hypothetical judgements a significance that goes much beyond questions of a uniform treatment of logical reasoning.

[6]The converse it not valid. For example, we cannot schematically derive $p \,\&\, q\ \mathsf{prop} \vdash p\ \mathsf{prop}$. This is due to the fact that inverses to the formation axioms are not available in SCML, i.e., we do not have $S(\underline{p}) \Rightarrow \mathbb{P}(S)$ as a axiom schema.

[7]Since we assume that $\langle\rangle \vdash \mathbb{P}(S)[\underline{A}]$ is derivable we have no *schematic* elimination of S. Thus the result is slightly weaker than the corresponding results of the previous chapters. The reason is again that we have no inverses of formation rules.

References

Aczel, P. (1980). *Frege structures and the notions of proposition, truth and set.* In: J. Barwise et al. (Eds.), *The Kleene Symposium.* Proceedings of the Symposium held June 18-24, 1978 at Madison, Wisconsin. Amsterdam: North-Holland (pp. 31-59).

Anderson, A.R. & Belnap, N.D. (1975). *Entailment: The Logic of Relevance and Necessity.* Vol. I. Princeton University Press.

Aronsson, M. & Montelius, J. (1986). *Generalized Horn-clauses, Interpreter.* SICS Technical Reports (Swedish Institute of Computer Science).

Avron, A. (1986). *Gentzenizing Schroeder-Heister's natural extension of natural deduction.* Manuscript (LFCS Edinburgh).

Beeson, M.J. (1985). *Foundations of Constructive Mathematics.* Metamathematical Studies. Berlin: Springer.

Belnap, N.D. (1982). *Display logic.* Journal of Philosophical Logic, 11, 375-417.

Došen, K. (1980). *Logical Constants: An Essay in Proof Theory.* D. Phil. thesis, University of Oxford.

Došen, K. (1985). *Sequent-systems for modal logic.* Journal of Symbolic Logic, 50, 149-168.

Došen, K. (1986a). *Higher-level sequent-systems for intuitionistic modal logic.* Publications de l'Institut Mathématique, 39 (53), 3-12.

Došen, K. (1986b). *Sequent-systems and groupoid models.* Manuscript (Mathematical Institute, Belgrade).

Došen, K. (1988). *Logical constants as punctuation marks.* Notre Dame Journal of Formal Logic (in press).

Došen, K. & Schroeder-Heister, P. (1985). *Conservativeness and uniqueness.* Theoria, 51, 159-173.

Došen, K. & Schroeder-Heister, P. (1987). *Uniqueness, definability and interpolation.* Journal of Symbolic Logic (in press).

Dummett, M. (1975). *The philosophical basis of intuitionistic logic.* In: H.E. Rose & J.E. Shepherdson (Eds.), *Logic Colloquium '73.* Proceedings of the Logic Colloquium, Bristol, July 1973. Amsterdam: North-Holland (pp. 5-40).

Dummett, M. (with the assistance of R. Minio) (1977) *Elements of Intuitionism.* Oxford: Clarendon Press.

Dunn, J.M. (1975). *Consecution formulation of positive R with co-tenability and t.* In: Anderson & Belnap (1975), 28.5 (pp. 381-391).

Fitch, F.B. (1952). *Symbolic Logic: An Introduction.* New York: Ronald Press.

Frege, G. (1893). *Grundgesetze der Arithmetik: Begriffsschriftlich abgeleitet.* I. Band. Jena: H. Pohle.

Gabbay, D.M. & Reyle, U. (1984). *N-PROLOG: An extension of PROLOG with hypothetical implications.* I. Journal of Logic Programming, 1, 319-355.

Gentzen, G. (1935a). *Untersuchungen über das logische Schließen.* Mathematische Zeitschrift, 39, 176-210, 405-431. English translation in M.E. Szabo (Ed.), *The Collected Papers of Gerhard Gentzen*, Amsterdam: North-Holland 1969 (pp. 68-131).

Gentzen, G. (1935b). *Die Widerspruchsfreiheit der reinen Zahlentheorie.* Mathematische Annalen, 112, 493-565. English translation in Szabo (Ed.), *Collected Papers* (pp. 132-201).

Gentzen, G. (1938). *Die gegenwärtige Lage in der mathematischen Grundlagenforschung.* Leipzig (= Forschungen zur Logik und zur Grundlegung der exakten Wissenschaften, Neue Folge, 4). English translation in Szabo (Ed.), *Collected Papers* (pp. 234-251).

Giambrone, S. (1983). *Gentzen Systems and Decision Procedures for Relevant Logics.* Ph. D. thesis. Australian National University, Canberra.

Hallnäs, L. (1986). *Generalized Horn clauses.* SICS Technical Report (Swedish Institute of Computer Science).

Hallnäs, L. & Schroeder-Heister, P. (1987). *A proof-theoretic approach to logic programming: I. Generalized Horn clauses* (submitted for publication).

Hallnäs, L. & Schroeder-Heister, P. (1988). *A proof-theoretic approach to logic programming: II. Program rules with quantifiers* (in preparation).

Harper, R., Honsell, F. & Plotkin, G. (1987). *A framework for defining logics.* Manuscript (LFCS Edinburgh).

Hilbert, D. (1923). *Die logischen Grundlagen der Mathematik.* Mathematische Annalen, 88, 151-165.

Howard, W.A. (1969/1980). *The formulae-as-types notion of construction.* In: J.P. Seldin & J.R. Hindley (Eds.), *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* London: Academic Press 1980 (pp. 479-490). Circulating as a manuscript since 1969.

Kutschera, F. v. (1968). *Die Vollständigkeit des Operatorensystems* $\{\neg, \wedge, \vee, \supset\}$ *für die intuitionistische Aussagenlogik im Rahmen der Gentzensemantik.* Archiv für mathematische Logik und Grundlagenforschung, 11, 3-16.

Kutschera, F. v. (1969). *Ein verallgemeinerter Widerlegungsbegriff für Gentzenkalküle.* Archiv für mathematische Logik und Grundlagenforschung, 12, 104-118.

Kutschera, F. v. (1985). *Der Satz vom ausgeschlossenen Dritten. Untersuchungen über die Grundlagen der Logik.* Berlin: de Gruyter.

Lloyd, J.W. (1984). *Foundations of Logic Programming.* Berlin: Springer.

Lorenzen, P. (1955). *Einführung in die operative Logik und Mathematik.* Berlin: Springer (2nd edition 1969).

Martin-Löf, P. (1975). *An intuitionistic theory of types: Predicative part.* In: H.E. Rose & J.C. Shepherdson (Eds.), *Logic Colloquium '73.* Proceedings of the Logic Colloquium, Bristol, July 1973. Amsterdam: North-Holland (pp. 73-118).

Martin-Löf, P. (1980). *Intuitionistic Type Theory.* Lectures at the meeting "Konstruktive Mengenlehre und Typentheorie" (München, Oktober 1980) (reported in Beeson 1985, Ch. XI).

Martin-Löf, P. (1982). *Constructive mathematics and computer programming.* In: L.J. Cohen et al. (Eds.), *Logic, Methodology and Philosophy of Science VI*, Amsterdam: North-Holland (pp. 153-175).

Martin-Löf, P. (1983). *Lectures on intuitionistic logic and type theory.* Università di Siena (tape transcription).

Martin-Löf, P. (1984). *Intuitionistic Type Theory.* Napoli: Bibliopolis.

Martin-Löf, P. (1985a). *On the meanings of the logical constants and the justifications of the logical laws.* In: *Atti degli incontri di logica matematica*, vol. 2. Università di Siena, Dipartimento di Matematica, Scuola di Specializzazione in Logica Matematica (pp. 203-281).

Martin-Löf, P. (1985b). *Truth of a proposition, evidence of a judgement, validity of a proof.* Lecture at the Workshop on Theories of Meaning, Centro Fiorentino di Storia e Filosofia della Scienza, Firenze, 3-7 June 1985.

Miller, D. (1986). *A theory of modules for logic programming.* IEEE Symposium on Logic Programming, Salt Lake City, Utah.

Nelson, D. (1949). *Constuctible falsity.* Journal of Symbolic Logic, 14, 16-26.

Prawitz, D. (1965). *Natural Deduction. A Proof-Theoretical Study.* Stockholm: Almqvist & Wiksell.

Prawitz, D. (1973). *Towards a foundation of a general proof theory.* In: P. Suppes et al. (Eds.), *Logic, Methodology and Philosophy of Science IV*, Amsterdam: North-Holland (pp. 225-250).

Prawitz, D. (1974). *On the Idea of a general proof theory.* Synthese, 27, 63-77.

Prawitz, D. (1977). *Meaning and proofs: On the conflict between classical and intuitionistic logic.* Theoria, 43, 2-40.

Prawitz, D. (1979/1982). *Proofs and the meaning and completeness of the logical constants.* In: J. Hintikka et al. (Eds.), *Essays on Mathematical and Philosophical Logic.* Dordrecht: Reidel 1979 (pp. 25-40). Revised German translation in Conceptus, 16, 1982, no. 38, 31-44.

Prawitz, D. (1981). *Philosophical aspects of proof theory.* In: G. Floistad (Ed.), *Contemporary Philosophy. A New Survey.* The Hague: Nijhoff (pp. 235-277).

Prawitz, D. (1985). *Remarks on some approaches to the concept of logical consequence.* Synthese, 62, 153-171.

Prawitz, D. (1987). *Dummett on a theory of meaning and its impact on logic.* In: B. M. Taylor (Ed.), *Michael Dummett: Contributions to Philosophy.* The Hague: Nijhoff.

Read, S. (1984). *Proof theory for relevant logics.* Manuscript (University of St. Andrews). Also part of a lecture at the Conference on Language and Logic, Leicester 1985.

Read, S. (1988). *Relevant Logic* (monograph). In preparation.

Robinson, J.A. (1965). *A machine-oriented logic based on the resolution principle.* Journal of the Association of Computing Machinery, 12, 23-41.

Routley, R., Plumwood, V., Meyer, R.K. & Brady, R.T. (1982). *Relevant Logics and their Rivals. Part I. The Basic Philosophical and Semantical Theory.* Atascadero, Calif.: Ridgeview Publishing Company.

Schroeder-Heister, P. (1981). *Untersuchungen zur regellogischen Deutung von Aussagenverknüpfungen.* Dr. phil. thesis, Universität Bonn.

Schroeder-Heister, P. (1982). *Sentential calculi with rules of arbitrary levels: Normalization and the completeness of the intuitionistic sentential operators.* Manuscript.

Schroeder-Heister, P. (1983). *The completeness of intuitionistic logic with respect to a validity concept based on an inversion principle.* Journal of Philosophical Logic, 12, 359-377.

Schroeder-Heister, P. (1984a). *A natural extension of natural deduction.* Journal of Symbolic Logic, 49, 1284-1300.

Schroeder-Heister, P. (1984b). *Generalized rules for quantifiers and the completeness of the intuitionistic operators* $\&$, $\vee$, $\supset$, $\curlywedge$, $\forall$, $\exists$. In: M.M. Richter et al. (Eds.), *Computation and Proof Theory.* Proceedings of the Logic Colloquium held in Aachen, July 18-23, 1983, Part II. Berlin: Springer Lecture Notes in Mathematics, Vol. 1104 (pp. 399-426).

Schroeder-Heister, P. (1985). *Proof-theoretic validity and the completeness of intuitionistic logic.* In: G. Dorn & P. Weingartner (Eds.), *Foundations of Logic and Linguistics: Problems and Their Solutions.* New York: Plenum Press (pp. 43-87).

Shoesmith, D.J. & Smiley, T.J. (1978). *Multiple-Conclusion Logic.* Cambridge University Press.

Slaney, J. (1984). *A natural deduction formulation of some relevant logics.* Manuscript (University of Edinburgh).

Slaney, J. (1986). *Curried Lemmon: A unified theory of relevant logics and their rivals.* Logic Colloquium '86, University of Hull.

Slaney, J. & Martin, R.L. (1988). *Contraction-free logics* (in preparation).

Smith, J. (1984). *An interpretation of Martin-Löf's type thory in a type-free theory of propositions.* Journal of Symbolic Logic, 49, 730-753.

Tarski, A. (1930). *Über einige fundamentale Begriffe der Metamathematik.* Comptes Rendus des séances de la Société des Sciences et des Lettres de Varsovie, vol. 23 (pp. 22-29). English translation in A. Tarski: *Logic, Semantics, Metamathematics. Papers from 1923 to 1938.* Ed. by J.H. Woodger, 2nd edition ed. by J. Corcoran. Indianapolis: Hackett 1983 (pp. 30-37).

Tennant, N. (1987). *Anti-Realism and Logic: Truth as Eternal.* Oxford: Clarendon Press.

Zucker, J.I. & Tragesser, R.S. (1978). *The adequacy problem for inferential logic.* Journal of Philosophical Logic, 7, 501-516.