Proof-Theoretic Semantics: Some Basic Ideas

with the manuscript

Proof-Theoretic and Constructive Consequence

Peter Schroeder-Heister
University of Tübingen

Preface

I publish here a previously unpublished manuscript, which was written during a sabbatical in the winter semester 2002-03 and completed in February 2003. Some passages of part I and part II have found their way into the article "Validity Concepts in Proof-Theoretic Semantics (Synthese 148, 2006, 525-571, doi:10.1007/s11229-004-6296-1). Attached is a manuscript of April 2003 which discusses the specific features of proof-theoretic consequence in relation to constructive consequence in general.

Tübingen, August 2022

Peter Schroeder-Heister

Contents

Unpublished Manuscript, written during a sabbatical in the winter semester 2002-03 and completed in February 2003. Some passages of part I and part II have found their way into the article "Validity Concepts in Proof-Theoretic Semantics (Synthese 148, 2006, 525-571, https://doi.org/10.1007/s11229-004-6296-1).

Proof-Theoretic Semantics: Some Basic Ideas

Peter Schroeder-Heister

Wilhelm-Schickard-Institut, Universität Tübingen
Sand 13, 72076 Tübingen, Germany
psh@informatik.uni-tuebingen.de

In this paper I describe and compare two basic approaches to what I call "proof-theoretic semantics". The "standard" approach, which is mainly due to Dummett and Prawitz, attempts to give a semantics of proofs by defining what counts as a valid proof. The second one, which is based on ideas by Hallnäs and the author, understands proofs semantically by reading the application of certain proof rules directly as semantical steps. Whereas the first one is a *global* approach, dealing with proofs as a whole and imposing requirements on them, the second one is *local* as is interprets individual proof steps without demanding from the very beginning that a proof composed of such single steps has special features.

Contents:

Introduction

Proof-theoretic semantics is an alternative to truth-condition semantics. It is based on the fundamental assumption that the central notion in terms of which meanings can be assigned to expressions of our language, in particular to logical constants, is that of proof rather than of truth. In this sense it is inherently inferential in spirit, as it is the inferential activity of human beings which manifests itself in proofs.

Proof-theoretic semantics has several roots, the most specific one being Gentzen's (1934) remarks that the introduction rules in his calculus of natural deduction define the meanings of logical constants while the elimination rules can be obtained as a consequence of this definition. More broadly, it belongs to the tradition according to which the meaning of a term has to be explained by reference to the way it is used in our language.

Although the "meaning as use" approach has been quite prominent for half a century now and provided one of the cornerstones of the philosophy of language, in particular of ordinary language philosophy, it has never become dominant in the *formal* semantics of artificial and natural languages. In formal semantics, the denotational approach which starts with interpretations of singular terms and predicates, then fixes the meaning of sentences in terms of truth conditions, and finally defines logical consequence as truth preservation under all interpretations, has always dominated. The main reason for this, as I see it, is the fact that, from the very beginning, denotational semantics received an authoritative rendering in Tarski's (1933) theory of truth, which combined philosophical claims with a sophisticated technical exposition and, at the same time, laid the ground for model theory as a mathematical discipline. Compared with this development, the "meaning as use" idea was a slogan supported by strong philosophical arguments, but without much formal backing.

There has been a lot of criticism of classical model-theoretic semantics from the denotational side itself. Examples are various theories rejecting the idea that *total* information about a model is available at every time, such as discourse representation theory, dynamical logics and partial logics (especially situation theory). Another example is Etchemendy's (1990) critique of classical consequence which gained much attention. However, in mainstream semantics there has never been a fundamental reorientation on the basis of the "meaning as use" idea which could have turned this idea into something like a formalized theory.

Proof-theoretic semantics, as a sidestream development, attempts to do just this. Naturally, it uses ideas from proof theory as a mathematical discipline, which is similar to the way truth-condition semantics relies on model theory. However, just this is the basis of a fundamental misconception of proof-theoretic semantics. A great deal of the development of mathematical proof theory has been dominated by the formalist reading

of Hilbert's program as dealing with *formal* proofs exclusively, in contradistinction to model theory as concerned with the (denotational) meaning of expressions. This dichotomy has entered many textbooks of logic in which "semantics" means model-theoretic semantics and "proof theory" just denotes the proof theory of formal systems. The result is that "proof-theoretic semantics" sounds like a contradiction in terms still today.

When I used this term since the late 1980s[1], it was not very common, although its content was there in the Swedish school of proof theory established by Prawitz and Martin-Löf. In the meantime it has gained some ground and there have been some occasional references to it (see the preface to this volume). Perhaps it will become more popular within general philosophy in the backwater of inferentialist approaches such as Brandom's[2], which more explicitly than ordinary language philosophy attempt to derive denotational meaning from inferential meaning, i.e., take the idea that meaning is rooted in proofs as their starting point.

Strictly speaking, the formalist reading of proof theory is not more foreign to the understanding of 'real' argumentation than model theory is to the interpretation of natural languages. In order to apply proof-theoretic results one has to consider formal proofs to be *representations* of proper arguments, just as, in order to apply model-theoretic methods, one has to consider formulas to be *representations* of proper sentences of a natural language like English. English is not *per se* a formal language, and arguments are not *per se* formal derivations. In this sense the term "proof-theoretic semantics" is as much and as little provocative as Montague's (1970) conception of "English as a formal language". Both proof-theoretic semantics and model-theoretic semantics are *indirect* in that they can only applied via a formal reading of natural language items. The basic difference is what these items are: proof-theoretic semantics starts from arguments and represents them by derivations, whereas model-theoretic semantics starts from names and sentences and represents them by individual terms and formulas.

As indicated above, it was the Swedish school in proof theory, which paved the way for a non-formalist philosophical understanding of proofs. Although originally dealing with problems of the proof-theory of formal systems, Prawitz and Martin-Löf realized soon that many of the concepts and methods developed there had a non-technical counterpart when looking at formal proofs as formal representation of "genuine" proofs. In taking Gentzen's remarks on the definitional significance of introduction and elimination rules seriously they developed cornerstones for proof-theoretic semantics.

---

[1] First in print in Schroeder-Heister (1991c).

[2] See especially Brandom (2000), where the relationship to Dummett's and Gentzen's approach is expressed very clearly.

An immediate predecessor of proof-theoretic semantics was Tait (1967), who, in his work on the *convertibility* of terms, developed concepts which are closely related to those later employed in proof-theoretic semantics. Another predecessor was Lorenzen (1955), who, in his *operative logics*, used arbitrary production rules as definitional rules from which, by means of an inversion principle, corresponding elimination rules can be obtained.

In the overview I am giving in this paper, I shall deal with basic technical concepts developed within proof-theoretic semantics. I shall not deal in much detail with the wider philosophical frameworks in which proof-theoretic semantics may be embedded. In particular, I shall not deal here with the "anti-realism" and "verificationism" in general, but just with technical constructs they end up with. The reason for this is lack of space, but also the fact indicated above that the desideratum of proof-theoretic semantics is not so much the general philosophical understanding but the formal development of the fundamental concepts. One result of this restriction is that I cannot give Dummett's work the attention it deserves, as his technical notions do not differ considerably from Prawitz's, though he has done enourmously much for the philosophical understanding of proof-theoretic semantics and for the fact that the general climate is now more in favour of this enterprise than it used to be.

I concentrate on two basic items: on *notions of proof-theoretic validity* as they have been developed mainly by Prawitz, and on *rule-based notions* proposed by Hallnäs and the author. Validity is a property of derivations, or more general "derivation structures", which are considered to be representations of arguments. The format of these derivations is Gentzen-style natural deduction. In the definitions of validity, attempts are made to justify arguments by turning certain proof-theoretic methods and results into semantical conditions, most prominently the following two: (1) Derivations can be simplified (or made more "direct") by certain reduction methods (terminating in *normal* derivations). (2) Assumption-free derivations in normal form are canonical (or "direct") in the sense that they apply an introduction rule in the last step. Valid arguments are then defined as derivation structures which exhibit properties like those found in standard natural deduction proofs such as (1) and (2). However, I shall strictly distinguish between genuine semantic features and technical properties used in normalization proofs. My main criticism of Prawitz will be that he does not sufficiently keep the semantical concepts apart from the technical concepts.

Rule-based notions do not try to justify whole proofs in the first place, but rather the individual rules whose applications make up a proof. This makes it possible to distinguish between features of rules and features of proofs generated by rules. These approaches have their origin (1) in proof-theoretic investigations on the general form of rules, in particular introduction and elimination rules for arbitrary logical constants, and (2) in ideas within logic programming concerning the definition of atomic formulas

by production rules. These ideas are turned into a rule-based theory of consequence with a theory of *definitional reflection* as its core which leads to a novel approach concerning introducing assumptions into formal reasoning: Assumptions cannot only be introduced in an unspecific way by just stating them, but also in a specific way depending on their form, or more general, on the way they are defined. This gives the sequent calculus a special philosophical significance. In the form of introduction rules on the left side of the sequent sign, i.e. in the antecedent, it embodies this idea of making assumptions depending on form and deductive context, in contradistinction to the unspecific way of making assumptions by means of initial sequents ($A \vdash A$). Definitional reflection may then be viewed as generalizing the idea of left-introduction rules of the sequent calculus. Among the applications of definitional reflection we only mention the analysis of circular reasoning as a simple but very instructive example that cannot be dealt with by validity approaches, which assume the well-foundedness of definitional rules from the very beginning. In particular, this example demonstrates how definitional reflection challenges features like cut elimination and normalization, which in more standard presentations are always postulated as something that should be provable under all circumstances.

I am aware that, in concentrating on these two bsic points, of which proof-theoretic validity is even treated in much greater detail, many basic ideas are omitted. The stimulations that have come to the field of proof-theoretic semantics from type theories, for example, are not mentioned at all. In particular, Martin-Löf's type theory, which is founded on strong semantical principles and may be correctly viewed as developing a whole programme of proof-theoretic semantics, is not dealt with. It is only briefly indicated in Part II how very elementary features of Martin-Löf's approach fit into the framework presented here. The discussion about classical vs. intuitionistic logic is completely left out, too. In taking positive implicational logic as our model, we stick to intuitionistic (or even minimal) logic. Categorical approaches to proof-theoretic semantics are not considered either. Even a rudimentary account of these items would make a substantial monograph out of this paper. The positive reasons for concentrating on notions of proof-theoretic validity and on rule-based notions is that they are based on very elementary principles and are very near to Gentzen's original programme of justifying logics. Another reason for dealing with Prawitz's notions of validity in some detail, thereby discussing and improving them in certain respects, is that there has not been a thorough investigation of his proposal within theories of meaning for logic so far.

Corresponding to the subjects chosen, this paper is divided into three parts: Part I deals with the background of proof-theoretic semantics. It starts with Gentzen's programme of justifying logical inferences and the way it is turned into an inversion principle by Prawitz. There follows a sketch of Lorenzen's approach in his "operative

logics", which may be considered a variant of proof-theoretic semantics in Gentzen's spirit, although it was never understood by Lorenzen as such. Finally, we discuss Belnap's constraints on semantical rules in his short paper of 1962, which has always figured prominently in discussions on how to define meanings in terms of proof rules. There is also an appendix which contains a brief presentation of Gentzen-style natural deduction, which is the formal framework for most of what follows.

Part II deals with Prawitz-style validity concepts for derivations. They are contrasted with concepts used in proofs of (strong) normalization, which were originally introduced by Tait and Martin-Löf. Special emphasis is put on the difference between these concepts and semantical concepts by calling those used for normalization "computability" and only the semantical ones "validity". Various forms of validity notions for implicational logic are defined and compared, among them notions of strict and strong validity which go beyond Prawitz's definitions but follow basic intuitions. These notions are then extended to general derivation structures with arbitrary reductions serving as justifications. The possibility of founding proof-theoretic semantics on elimination rather than introduction inferences is briefly discussed. Martin-Löf's approach also mentioned as one which puts the metalogical perspective of validity notions into the system itself.

Finally, Part III presents some ideas of rule-based approaches. It covers two basic items: general rules for logical constants based on the idea of the common content of systems of rules, and definitional reflection as an idea which takes the introduction of assumptions based on definitional considerations seriously, quite in coincidence with the above-mentioned basic conceptual features of the sequent calculus, and with the definitional reading of logic programming. Definitional reflection is considered a fundamental alternative to the "standard" approach by Prawitz. It is the approach for which I have most sympathy and which will be elaborated in further joint work with Hallnäs. A discussion of circular reasoning within the framework of definitional reflection concludes this paper.

Part I. The background of proof-theoretic semantics

I.1 Gentzen's programme and Prawitz's inversion principle

Proof-theoretic semantics in the sense discussed in this paper goes back to certain programmatic remarks in Gentzen's *Investigations into Natural Deduction*, where he gives a semantical interpretation of his inference rules[3]. By "goes back to" I mean that proof-theoretic semantics is either intentionally and explicitly related to Gentzen's remarks (such as Prawitz's approaches), or can be related to them for objective reasons, even if there is no explicit reference and perhaps no awareness of them (such as Lorenzen's programme).

Gentzen's remarks deal with the relationship between introduction and elimination inferences in natural deduction.

> The introductions represent, as it were, the 'definitions' of the symbols concerned, and the eliminations are no more, in the final analysis, than the consequences of these definitions. This fact may be expressed as follows: In eliminating a symbol, we may use the formula with whose terminal symbol we are dealing only 'in the sense afforded it by the introduction of that symbol'. (Gentzen 1934, p. 80)

This cannot mean, of course, that the elimination rules are *deducible* from the introduction rules in the literal sense of the word; in fact, they are not. It can only mean that they can be *justified* from them in some way.

> By making these ideas more precise it should be possible to display the E-inferences as unique functions of their corresponding I-inferences, on the basis of certain requirements. (Gentzen 1934, p. 81)

So the idea underlying Gentzen's programme is that we have "definitions" in the form of introduction rules and some sort of semantical reasoning which, by using "certain requirements", validates the elimination rules.

As indicated in the introduction, I shall not discuss in detail the philosophical reasons which might support Gentzen's programme, in particular the asymmetry between introduction and elimination rules. For that I would in particular have to refer to Dummett's work and his claim that there are two different aspects of language use: one connected with 'directly' or 'canonically' asserting a sentence, and another one with drawing consequences from an assertion.[4] The first is the primary or 'self-justifying'

[3]For Gentzen's system of natural deduction and the terminology and notation used in this paper see the appendix to this part (I.4).

[4]See especially Dummett (1991).

way corresponding to reasoning with introduction rules, whereas the second one, which corresponds to reasoning with elimination rules, is in need of justification. This justification relies on the *harmony* required to hold between both aspects: The possible *consequences* to be drawn from an assertion are determined by the *premisses* from which the assertion can possibly be inferred by direct means. Dummett's approach is a very general account of language use. It tries to give the 'meaning as use'- slogan a constructive rendering without the holistic connotations it often has, according to which language can only be interpreted as a whole. As Dummett correctly argues, holism lacks explanatory power and is unsuitable for formal investigations.

Prawitz, in an "inversion principle"[5] formulated in his classic monograph on *Natural Deduction* of 1965, tried to make Gentzen's remarks more precise.

> Let α be an application of an elimination rule that has B as consequence. Then, deductions that satisfy the sufficient condition [...] for deriving the major premiss of α, when combined with deductions of the minor premisses of α (if any), already "contain" a deduction of B; the deduction of B is thus obtainable directly from the given deductions without the addition of α. (Prawitz 1965, p. 33)

Here the sufficient conditions are given by the premisses of the corresponding introduction rules. Thus the inversion principle says that a derivation of the conclusion of an elimination rule can be obtained without an application of the elimination rule if its major premiss has been derived using an introduction rule in the last step, which means that a combination

$$\dfrac{\dfrac{\mathcal{D}}{A}\ \text{I inference} \qquad \{\mathcal{D}_i\}}{B}\ \text{E inference}$$

of steps, where $\{\mathcal{D}_i\}$ stands for a (possibly empty) list of deductions of minor premisses, can be removed.

At first sight this just states the fact that maximum formulas, i.e., formulas being conclusions of an I inference and at the same time major premiss of an E inference (in the example: A), can be eliminated by certain reductions, thus leading to the idea of normal derivations[6]. However, it also represents a *semantical* interpretation of elimination inferences by saying that nothing is gained by an application of an elimination rule if its major premiss has been derived *according to its meaning* (i.e., by means of an introduction rule). So the main reductions given by Prawitz in connection with normalization are at the same time semantical justifications of E rules with respect to

[5] Named following Lorenzen. See the next section (I.2).

[6] See the appendix (I.4).

I rules. His inversion principle just elaborates Gentzen's idea of "special requirements" needed for this justification, by demanding that E rules invert I rules in a precise sense.

That it corresponds indeed to what Gentzen had in mind can be seen from the only example Gentzen gives:

> We were able to introduce the formula $A{\to}B$ when there existed a derivation of B from the assumption formula A. If we then wished to use that formula by eliminating the $\to$-symbol (we could, of course, also use it to form longer formulae, e.g., $(A{\to}B){\vee}C$, $\vee$-I), we could do this precisely by inferring B directly, once A has been proved, for what $A{\to}B$ attests is just the existence of a derivation of B from A. (Gentzen 1934, pp. 80–81)

This may be read as follows: Given the situation

$$\dfrac{\dfrac{\begin{array}{c}A\\ \mathcal{D}\\ B\end{array}}{A{\to}B}\quad\begin{array}{c}\mathcal{D}'\\ A\end{array}}{B}$$

where $\mathcal{D}$ is "a derivation of B from the assumption formula A", and $\mathcal{D}'$ is the derivation showing that "A has been proved", so that we can use $A{\to}B$ to obtain B "by eliminating the $\to$-symbol". Then by means of

$$\begin{array}{c}\mathcal{D}'\\ A\\ \mathcal{D}\\ B\end{array}$$

we can infer "B directly, once A has been proved [by means of $\mathcal{D}$']", as "$A{\to}B$ attests [...] the existence of a derivation [viz. $\mathcal{D}$] of B from A".

However, although Gentzen's remarks are correctly read as outlining a semantical programme, he himself takes a more formalistic stance, which is clear from his writings in general and from the sentence immediately following the quoted passage:

> Note that in saying this we need not go into the "informal sense" ["inhaltlicher Sinn"][7] of the $\to$-symbol. (Gentzen 1934, p. 81)

Prawitz (1965) deserves the credit to have drawn our attention to the genuine semantical content of Gentzen's remarks, though this is not spelled out in detail in his monograph. Only later in Prawitz (1971) and in particular in Prawitz (1973, 1974) it is turned into a full-fledged semantical theory.

[7] Quotes by Gentzen

In the following, when giving examples, we confine ourselves to implication, just as Gentzen did, so we are dealing with positive implicational logic as our model. This simplification is justified for the reason that a proper treatment of implication gives indeed a strong guideline for the proper treatment of other logical operators. Implication is the most complicated propositional operator, which shares certain properties especially with universal quantification, as the distinction between open and closed derivations, which will turn out as semantically crucial, is to a great extent due to its presence.

I.2 Lorenzen's operative semantics

Although Lorenzen's *Introduction to Operative Logics and Mathematics* (1955) is formalistic in spirit, basing logical and mathematical reasoning on 'operating' with symbolic figures (in this sense being related to Curry's approach), it contains a semantics for the logical constants even if not intended and designated as such. In dealing with logical constants, Lorenzen is concerned with the *justification* of the inference rules governing them, arriving at a formalism of intuitionistic first-order logic. This justification is a sort of proof-theoretic semantics in our sense as it establishes the *correct use* of those constants by *reflecting on proofs* and not, as one would expect from a formalist, by just laying down the set of intended inference rules. To be sure, the proofs Lorenzen is reflecting upon are formal proofs, i.e. derivations, and the logical rules being justified are formal rules as well. However, the way through which he arrives at these logical rules is semantical reasoning in our sense of the word, based on principles for the validation of rules. In fact, a crucial rule is played by an *inversion principle*, which similar to Prawitz's one justifies elimination inferences from introduction inferences. It was actually Lorenzen who coined the term *inversion principle* for a more general claim not confined to logical constants, which was then later borrowed by Prawitz for use in the context of natural deduction.

Lorenzen starts with logic-free (atomic) calculi, which correspond to production systems or grammars in different terminologies. He calls a rule *admissible* with respect to such a system if it can be added to it without enlarging the set of its derivable atoms[8]. Therefore, if $\vdash_K$ denotes derivability in a calculus K, then the rule $a_1, \ldots, a_n \to a$ is admissible, if $\vdash_K a_i$ for all i $(1 \leq i \leq n)$ implies $\vdash_K a$, to be distinguished from the derivability of a *from the assumptions* $a_1, \ldots, a_n$, which is denoted by $a_1, \ldots, a_n \vdash_K a$.

The implication arrow $\to$ is identified with the rule arrow used in stating a production rule, and the derivability of an implication is interpreted as an admissibility statement. If $A, B_1, \ldots, B_n$ stand for lists of atoms and $a, b_1, \ldots, b_n$

[8]In contradistinction to Lorenzen, and following the terminology of logic programming, by *atoms* I denote the formulas of an atomic system, i.e., its *words* in the terminology of grammars.

are atoms, then $\vdash_K A \to a$ means that the rule $A \to a$ is admissible in K, and $B_1 \to b_1, \ldots, B_n \to b_n \vdash_K A \to a$ expresses that the rule $A \to a$ is admissible in the calculus which results from K by adjoining $B_i \to b_i$ $(1 \leq i \leq n)$ as additional rules. For iterated implications such as $(A \to b) \to a$ Lorenzen develops a theory of admissibility statements of higher levels, which cannot be presented here[9].

Since certain statements such as $\vdash_K a \to a$ or $a \to b, b \to c \vdash_K a \to c$ hold independently of the underlying system K (they are called *universally admissible* ["allgemeinzulässig"]), Lorenzen obtains a set of general principles which constitute a system of positive implicational logic.

This may already be considered to be a proof-theoretic semantics for implication. Implications express admissibility statements with respect to formal systems, and logical implications express admissibility statements which hold for *any* atomic system. In a related way, laws for universal quantification $\forall$ are justified using universal admissibility statements for rules of the form $A \to_{x_1,\ldots,x_n} a$ with schematic variables $x_1, \ldots, x_n$ as indices (with premiss-free rules $\to_{x_1,\ldots,x_n} a$ as a limiting case).

However, in our context, Lorenzen's justification of logical constants like $\wedge$, $\vee$ and $\exists$ is even more interesting, as it is closely related to all other approaches dealt with here, which are based on Gentzen's ideas. Lorenzen proceeds by deducing a list of principles for establishing admissibility, the crucial one being his *inversion principle*. In a very simplified form, without taking variables in rules into account, this inversion principle says the following. Let $A, A_1, \ldots, A_n$ be lists of atoms, and a, b atoms. Suppose

$$\begin{array}{lcl} A_1 & \to & a \\ & \vdots & \\ A_n & \to & a \end{array}$$

are the only rules by means of which a can be derived in a calculus K. Then the rule $a \to c$ is admissible in the calculus which results from K by adjoining

$$\begin{array}{lcl} A_1 & \to & c \\ & \vdots & \\ A_n & \to & c \end{array}$$

as primitive rules. Roughly speaking: Everything that is derivable from each condition of a follows from a itself. This principle is used for the justification of logical inference rules as follows:

Let *disjunction* be defined by a pair of rules

$$\begin{array}{lcl} a & \to & a \vee b \\ b & \to & a \vee b \end{array}$$

[9] and which is not without serious problems

Then the inversion principle says that $a \vee b \to c$ is admissible if both $a \to c$ and $b \to c$ are admissible ($a \to b, b \to c \vdash a \vee b \to c$), which justifies the elimination rule for disjunction.

Similarly, for *conjunction*, which is defined by

$$a, b \to a \wedge b$$

Then, by the inversion principle, we can argue that $a \wedge b \to c$ is admissible if $a, b \to c$ is admissible. Since $a, b \to a$ and $a, b \to b$ are trivially admissible, this implies that both $a \wedge b \to a$ and $a \wedge b \to b$ are admissible, which justifies the elimination rules for conjunction.

Using rules with bound variables, from the rule

$$a \to \exists x a$$

the elimination rule for *existential quantification* can be justified in the form of the principle

$$a \to_x c \vdash \exists x a \to c$$

(for c not containing x free) in an analogous way.

Finally, the intuitionistic *absurdity rule* is justified by reference to the admissibility of

$$\bot \to c$$

with respect to any calculus in which $\bot$ is undefined, i.e., in which $\bot$ is not the head of a primitive rule. This might be considered to be a limiting case of the inversion principle (with respect to an empty set of defining rules). However, Lorenzen formulates it as based on a principle of its own, called the *underivability principle* ["Unableitbarkeitsprinzip"].

It is obvious that this justification of logical rules is very closely related to Gentzen's programme of taking introduction rules as definitions of constants and justifying elimination rules with respect to introduction rules by assuming, in each case, that the major premiss of the elimination rule has been derived using one of the introduction rules. It is a similar sort of reasoning, upon which the validity of Lorenzen's inversion principle is based: In order to show that $a \to c$ is admissible we have to show that c can be derived, given that a has been derived. "Given that a has been derived" means that one of the rules $A_i \to a$ available as defining rules for a has been used in the last step, which means that A_i must have been derived for some i. This means that c can be derived, if all rules of the form $A_i \to c$ are added as primitive rules.

Of course, this is not exactly what Gentzen had in mind, notably with respect to implication. When basing his notion of implication on the concept of admissibility of rules, Lorenzen relies on a system crucially different from Gentzen's calculus of natural deduction in that it does not employ the idea of discharging assumptions when

introducing implication. However, the justifications of the rules for all other connectives do not so much differ from Gentzen's.

In other respects Lorenzen's approach is much more general than Gentzen's in that he considers logically compound sentences just as special cases of arbitrary atoms. This means that his inversion principle can be used as a justification of elimination rules from introduction rules for arbitrary atoms. This makes his approach capable of dealing with inductive definitions in general rather than just with introduction and elimination rules for logical constants. He thus anticipates the idea of introduction and elimination rules for atoms which can be found, e.g., in Martin-Löf's (1971) theory of iterated inductive definitions[10] and in- general reflection principles for logic programming. In fact, his inversion principle in its general form is closely related to a certain form of the principle of definitional reflection (see section III.3.2 below).

I.3 The Belnap conditions and other constraints for semantical rules

The most primitive form of the "meaning as use" paradigm with respect to logical operators is the claim that an arbitrary set of rules of which form whatsoever implicitly determines the meaning of logical operators occurring in them. In this sense we would just have to stipulate certain rules to hold of an operator in order to equip it with meaning. Its *holistic* version would be that several constants may be defined by one and the same set of rules, whereas a *constructive* variant expects that such a set of meaning-giving rules must be separable into disjoint subsets which can be well-ordered in the following way: Every subset defines the meaning of one particular constant, where the meanings of other constants used in the definition are defined by rule sets preceding this set in the well-ordering. The latter condition will also be called the *well-foundedness requirement*.

However, neither the holistic nor the constructive version of this approach will be considered a proof-theoretic semantics in the genuine sense. For semantical purposes it is not sufficient just to lay down rules. As Prior (1960) has shown, admitting arbitrary rules as meaning rules can easily result in implausible, and in the worst case inconsistent, results, even though the constructive well-foundedness requirement is met. As an example Prior gives the combination of the $\vee$-introduction with the $\wedge$-elimination rules in the form of an operator called *tonk*:

$$\frac{A}{A \textit{ tonk } B} \quad \frac{B}{A \textit{ tonk } B} \qquad \frac{A \textit{ tonk } B}{A} \quad \frac{A \textit{ tonk } B}{B}$$

[10] In contradistinction to Martin-Löf, Lorenzen considers the induction principle as a principle of its own rather than an application of the inversion principle. Martin-Löf subsumes the induction rules under the generalized notion of an elimination rule.

allowing one to derive any B from any A. Therefore, there have to be restrictions on rules if they should qualify as meaning-giving rules in a reasonable way.

Belnap (1962) proposed to require *conservativeness* and *uniqueness* in the following sense. The meaning-giving rules for a constant α should not allow one to prove something which can be formulated in the vocabulary available prior to the definition of α (conservativeness). Furthermore, α should be characterized in such a way that, if we duplicate its rules by adding identical rules for some α^*, any formula $A(\alpha)$ containing α (but not α^*) should be interdeducible with the formula $A(\alpha^*)$, in which α is replaced with α^* (uniqueness). Conservativeness is weaker than eliminability of definienda by definientia which, besides uniqueness, is normally expected to hold of explicit definitions. For example, it also covers inductive definitions.

It is tempting to base proof-theoretic semantics on the Belnap criteria of conservativeness and uniqueness. However, according to my understanding of proof-theoretic semantics, these criteria are too wide in one respect and to narrow in another, if one wants to delineate semantical rules. Of course, meaning-giving rules should be conservative in any case. However, uniquess is too narrow a criterion. Later on I shall consider *relative* notions of uniquess of the sort "The definiendum is uniquely defined *given* the definiens is unique, even if the definiens cannot be proven unique". This makes it possible to also investigate partial notions of meaning.

On the other hand, conservativeness and uniqueness are also too wide to justify rules as meaning-giving, as they do not make any restriction concerning the *form* of rules. In all variants of proof-theoretic semantics presented in the following, the constant whose meaning is to be explained, occurs in a special position, normally as the *conclusion* of a rule, which may be a production rule in an atomic system, a definitional (program) rule in theories related to logic programming, or an introduction rule in natural deduction. In a dual approach based on eliminations, the constant may occur in *premiss* position. But again, it occurs at a special place. I call this the *special form requirement*. Furthermore, the constant in question occurs as the major operator of a proposition, i.e. is not embedded within other connectives or quantifiers, not even with the defined operator itself (i.e., for defining conjunction $\wedge$, we do not allow rules containing iterated conjunctions of the form $A\wedge(B\wedge C)$ etc. This will be called the *explicitness requirement*. The *special form* and *explicitness* requirements make proof-theoretic semantics in our sense strongly differ from rule systems which just guarantee conservativeness and uniqueness.

This does not mean that proof-theoretic semantics has to be *constructive* in the sense described above. What has to be required is *separativity*, i.e., the condition that semantical rules give meaning to *single* constants rather than to several constants at the same time, but not well-foundedness, which is characteristic of a constructive position. Therefore, the *special form* and *explicitness* requirements are specifications

of the *separativity* constraint, not part of a very restrictive constructivity option. In any case these requirements are only necessary, not sufficient. They can be used to exclude certain systems of rules as inappropriate. But the validation of inference rules with respect to certain semantical principles and ideas is still the major task of systems of proof-theoretic semantics.

As mentioned above, conservativeness should hold in any case. However, in natural deduction or sequent systems satisfying the separativity, special form and explicitness constraints, conservativeness can be expressed by more specific properties which refer to the exact form of rules considered. For example, the features of eliminability of cut or of normalization may replace the conservativeness option in such systems. By considering different notions of cut elimination or normalization, more fine-grained notions can be considered). One possibility I shall deal with in section III.3.2 is *relative cut*: Cut holds for a definiendum given it holds for its definiens, without necessarily being provable for the definiens[11].

So our general picture is the following: For meaning explanations we expect *separativity* (different rules for different constants) to hold, and furthermore *special form of rules* and *explicitness* (constants at certain positions in defining rules, and not nested). Conservativeness has to hold, but will be investigated in close relationship with notions of cut elimination and normalization. Constructivity (well-foundedness of sets of definitional rules) is not required, but may of course be studied as a property of particular systems.

I.4 Appendix: Gentzen-style natural deduction

Gentzen's calculus of natural deduction and its classical rendering by Prawitz (1965) is the background to most topics in this paper. As the philosophical reader might not always be familiar with this particular form of natural deduction, I give a short sketch of some major features of the system and also fix my terminology. Besides Gentzen's (1934) original presentation and Prawitz's (1965) monograph, Tennant (1978) and Negri/von Plato (2001b) can be recommended as references.

I always use the term "derivation" for formal proofs in formal systems. When no further specification is given, I reserve "proof" and "deduction" for the case when 'genuine' arguments containing content are concerned. This differs in part from the terminology used in the sources above.

Formulas of first-order logic are written in the usual form. Concerning substitution, $A(x)$ denotes a formula in which x may occur free, and, when occurring in the same

[11] This does not mean that the global meaning of conservativeness, viz. that nothing new is provable in the old vocabulary, is given up. In systems with cut, the eliminability of cut guarantees conservativeness. In systems without the cut rule, conservativeness holds anyway.

context, $A(t)$ denotes the result of substituting t for x in $A(x)$, presupposing that this does not lead to variable confusions.

Natural deduction in general is based on two major ideas:

1. Derivations are always derivations from assumptions. Assumptions can be "discharged" or "eliminated" in the course of a derivation.

2. The rules for logical constants come in pairs. The introduction rule(s) allow(s) one to infer a proposition with the constant in question as the main operator, the elimination rule(s) permit(s) to draw consequences from such a proposition.

In Gentzen's natural deduction system for first order logic derivations are written in tree form and based on the following rules:

$$\frac{A \quad B}{A\wedge B} \qquad \frac{A\wedge B}{A} \quad \frac{A\wedge B}{B}$$

$$\frac{A}{A\vee B} \quad \frac{B}{A\vee B} \qquad \frac{A\vee B \quad \begin{matrix}[A]\\ C\end{matrix} \quad \begin{matrix}[B]\\ C\end{matrix}}{C}$$

$$\frac{\begin{matrix}[A]\\ B\end{matrix}}{A\to C} \qquad \frac{A\to B \quad A}{C}$$

$$\frac{\bot}{A}$$

$$\frac{A(y)}{\forall x A(x)} \qquad \frac{\forall x A(x)}{A(t)}$$

$$\frac{A(t)}{\exists x A(x)} \qquad \frac{\exists x A(x) \quad \begin{matrix}[A(z)]\\ C\end{matrix}}{C}$$

where the *eigenvariable* y is not free in any assumption, on which $A(y)$ depends, and the eigenvariable z is not free in any assumption exept the displayed assumption $A(z)$.

The rules in the left column are the *introduction rules* (*I rules*) for the operator displayed below the inference line, and the rules in the right column are the *elimination rules* (*E rules*) for the operator displayed above the inference line. The leftmost premiss in an elimination inference is called its major premiss, whereas the other premisses are called its minor premisses. Assumptions which can be discharged at the application of the rule in question are indicated by square brackets.

The system considered is one for intuitionistic logic, not classical logic. For classical logic one would have to add some rule or axiom which does not fit neatly within the pattern given above, such as the *tertium non datur* $A\vee\neg A$ or the double negation rule

$$\begin{array}{c}[\neg\neg A]\\ \vdots\\ \dfrac{\bot}{A}\end{array}\ .$$

For a proper treatment of classical logic within proof-theoretic semantics, other tools have to be employed, which go beyond the scope of the present paper (but see Tait's contribution to this volume).

Following Prawitz, I shall use the following notation: If a derivation $\mathcal{D}$ ends with A, I also write $\begin{array}{c}\mathcal{D}\\ A\end{array}$, if it depends on an assumption B, I also write $\begin{array}{c}B\\ \mathcal{D}\end{array}$ or $\begin{array}{c}B\\ \mathcal{D}\\ A\end{array}$. This means that the notations $\mathcal{D}$, $\begin{array}{c}\mathcal{D}\\ A\end{array}$, $\begin{array}{c}B\\ \mathcal{D}\end{array}$ and $\begin{array}{c}B\\ \mathcal{D}\\ A\end{array}$ do not denote different derivations, but just differ in what they make explicit. The *open assumptions* of a derivation are the assumptions on which the end-formula depends. The *open variables* of a derivation are those free individual variables which are not used as *eigenvariables* further down in the derivation tree. A derivation is called *closed*, if it contains either no open assumptions or open variables, otherwise it is called *open*. To indicate that x is open in $\mathcal{D}$, we also write $\mathcal{D}(x)$, and correspondingly $\mathcal{D}(t)$ for the result of substituting t for x throughout $\mathcal{D}$.

In his monograph of 1965, Prawitz's enterprise was to advance Gentzen's view that natural deduction captures logical reasoning in a most fundamental way. He showed that it has sophisticated matalogical features, notably normalization, which gives it a proof-theoretic significance corresponding to that of the the sequent calculus which so far had been considered the main system developed by Gentzen and practically exclusively studied in proof theory.

A derivation is said to be in normal form, if it does not contain deviation or detours in the sense that an operator is introduced by an introduction inferences and eliminated by an elimination inference. So-called *maximum formulas*, which are conclusions of introduction inferences and at the same time major premisses of elimination inferences

are removed by using the following *main reductions* of derivations:

$$\dfrac{\dfrac{\mathcal{D}_1 \quad \mathcal{D}_2}{A_1 \quad A_2}}{\dfrac{A_1 \wedge A_2}{A_i}} \qquad \text{reduces to} \qquad \begin{matrix}\mathcal{D}_i \\ A_i\end{matrix} \quad (i = 1, 2)$$

$$\dfrac{\dfrac{\mathcal{D}}{A_i} \quad \begin{matrix}[A_1] \\ \mathcal{D}_1 \\ C\end{matrix} \quad \begin{matrix}[A_2] \\ \mathcal{D}_2 \\ C\end{matrix}}{A_1 \vee A_2} \qquad \text{reduces to} \qquad \begin{matrix}\mathcal{D} \\ A_i \\ \mathcal{D}_i \\ \mathrm{C}\end{matrix} \quad (i = 1, 2)$$

$$\dfrac{\dfrac{\begin{matrix}A \\ \mathcal{D} \\ B\end{matrix}}{A \to B} \quad \begin{matrix}\mathcal{D}' \\ A\end{matrix}}{B} \qquad \text{reduces to} \qquad \begin{matrix}\mathcal{D}' \\ A \\ \mathcal{D} \\ B\end{matrix}$$

$$\dfrac{\begin{matrix}\mathcal{D}(y) \\ A(y)\end{matrix}}{\dfrac{\forall x A(x)}{A(t)}} \qquad \text{reduces to} \qquad \begin{matrix}\mathcal{D}(t) \\ A(t)\end{matrix}$$

$$\dfrac{\dfrac{\mathcal{D}}{A(t)} \quad \begin{matrix}[A(z)] \\ \mathcal{D}'(z) \\ C\end{matrix}}{\exists x A(x)} \qquad \text{reduces to} \qquad \begin{matrix}\mathcal{D} \\ A(t) \\ \mathcal{D}'(t) \\ \mathrm{C}\end{matrix}$$

In addition, there is the possibility of maximum *segments*, which consist of sequences of identical formulas in a branch of a derivation, beginning with a conclusion of an I rule, which is at the same time a minor premiss of an E rule, then consisting of conclusions of E rules which are at the same time minor premisses of E rules, and ending with a major premiss of an E rule. This means that maximum segments are sequences of C's in $\vee$or $\exists$elimination rules, starting with the conclusion of an I rule and ending with the major premiss of an E rule). They are reduced to maximum formulas using the so-called *permutative reductions*. They are important for normalization in the presence of disjunction and existential quantification. However, we do not consider them in detail as their semantical interpretation is limited (or at least goes beyond the scope of this paper).

Prawitz then shows that by iterated application of reduction steps, every derivation in intuitionistic logic can be normalized, i.e., can be rewritten to a derivation in normal form. One fundamental corollary of this result is that every closed derivation in intuitionistic logic can be reduced to one using an introduction rule in the last step, as a closed normal derivation is of exactly that form.

The normalization result mentioned is also called *weak normalization*. The *strong* normalization result says that any reduction sequence terminates in a normal derivation, no matter in which order reduction steps are applied. The methods used to prove strong normalization are strongly related to concepts used in proof-theoretic semantics and are mentioned in the text, as far as they are semantically relevant.

Most of the investigations in this paper will be carried out only for positive implicational logic, called $\mathcal{L}$. The language of this system just contains implicational formulas over propositional variables, and its inference rules are the introduction and elimination rules for implication.

Part II. Proof-theoretic validity

II.1 Normalization, computability and validity

II.1.1 Normalization and computability

Normalization plays a prominent role in the formal background of proof-theoretic semantics, in particular the result that normal closed proofs of complex formulas are in introduction form, i.e., use an introduction inference in the last step (see section I.4).

Of equal importance was the development of a technical method within normalization theory, which is used especially in proofs of *strong* normalization. By means of this method a certain predicate P of proofs is defined, which has the property that it entails (strong) normalizability. The predicate P has some flavour of a semantical predicate, and in a kind of correctness proof it can be shown that every derivation satisfies P, yielding as a corrollary that every derivation is (strongly) normalizable. Such a predicate was first defined by Tait (1967) under the name "convertibility" and used to demonstrate (weak) normalizability of terms. Martin-Löf (1971) carried his idea over from terms to derivations and defined a corresponding predicate which he called "computability", proving (weak) normalization for an extension of first-order logic, called the theory of iterated inductive definitions. At the same time, Girard (1971) used it to prove (weak) normalization for second-order logic. Again at the same time, it was Prawitz (1971) who emphasized its particular usefulness for proving *strong* normalization, calling it "strong validity". Since then it has been the basis of proofs of strong normalization for a variety of systems.[12]

In the following I shall speak of *computability* predicates or *the computability* predicate when dealing with this notion as it is used in normalization proofs, therefore relying on Martin-Löf's terminology. The term "valid" will be reserved for genuinely semantical notions applied in proof-theoretic semantics. I consider Prawitz's terminology, who speaks of "validity based on the introduction rules" (1971, p. 284) in contradistinction to "validity used in proofs of normalizability" (1971, p. 290) somewhat unfortunate, as there are strong differences between these notions, which appear to be more important than what they have in common. That there is a difference in principle between these terms is actually one of my basic claims.

I restrict Prawitz's notion of computability (called by him "validity used in proofs of normalizability") to positive implicational logic $\mathcal{L}$, i.e., to the system with only introduction and elimination rules for implications as primitive rules of inference. Under this restriction, Prawitz's computability notion is basically the same as Martin-Löf's.

[12] As a recent paper with many references to the literature see Joachimski & Matthes (2003).

A derivation is in *I-form* if it uses an introduction rule in the last step, i.e., if it is of the form

$$\dfrac{\begin{array}{c}[A]\\ \mathcal{D}\\ B\end{array}}{A{\to}B}.$$

Using a terminology which goes back to Dummett and common in proof-theoretic semantics, such a derivation is also called *canonical*. Let $\mathcal{D} \succ_1 \mathcal{D}'$ mean that the derivation $\mathcal{D}$ reduces to the derivation $\mathcal{D}'$ by applying a single reduction step to a subderivation of $\mathcal{D}$.

Definition of computability

(i) A derivation of the form $\dfrac{\begin{array}{c}[A]\\ \mathcal{D}\\ B\end{array}}{A{\to}B}$ *is computable, if for every computable* $\begin{array}{c}\mathcal{D}'\\ A\end{array}$ *,*

$\begin{array}{c}\mathcal{D}'\\ A\\ \mathcal{D}\\ B\end{array}$ *is computable.*

(ii) If a derivation $\mathcal{D}$ is not in I-form and is normal, then it is computable.

(iii) If a derivation is not in I-form and is not normal, then $\mathcal{D}$ is computable if, for every $\mathcal{D}'$ such that $\mathcal{D} \succ_1 \mathcal{D}'$, $\mathcal{D}'$ is computable.

This is a generalized inductive definition. It uses induction over the degree of the end formula of the derivation (clause (i)), and, within each degree, induction over the reducibility relation[13] (clauses (ii) and (iii)).

The proof of strong normalization then proceeds be establishing the following two propositions:

Proposition 1 *Every computable derivation is strongly normalizable.*

Proposition 2 *Every derivation is computable.*

Proposition 1 is a (nearly) immediate consequence of the definition of computability. Proposition 2 is based on a kind of correctness proof, verifying step by step that clauses (i) to (iii) of this definition are carried over from the premisses to the conclusion of an inference step. Other formulations of "computability" differ slightly from the one

[13] i.e., induction given by the operator associating with a set of derivations X of a formula the set of those derivations which reduce in one step to a derivation in X

given here. However, the basic features remain the same. The resulting normalization proofs all proceed via Propositions 1 and 2.

Computable derivations are closed under substitution with computable derivations, i.e., the following lemma holds:

Substitution lemma for computability

If $\begin{array}{c} A_1 \ldots A_n \\ \mathcal{D} \\ B \end{array}$ *is computable, where all open assumptions of* $\mathcal{D}$ *are among* $A_1, \ldots, A_n$,

then for any list of computable derivations $\begin{array}{c} \mathcal{D}_i \\ A_i \end{array}$ $(1 \leq i \leq n)$,

$\begin{array}{c} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \ldots A_n \\ \mathcal{D} \\ B \end{array}$ *is computable.*

Note that the converse direction of the lemma is trivial, as every assumption A_i is itself a normal, and therefore computable, derivation of A_i from A_i.

If one denotes closure under substitution with computable derivations as *computability under substitution*, one might formulate the lemma as saying that computability implies computability under substitution.

Weaker versions of computability just entail (weak) normalization. Instead of requiring in clause (iii) that *every* $\mathcal{D}'$ such that $\mathcal{D} \succ_1 \mathcal{D}'$, $\mathcal{D}'$ be computable, one might demand that a certain $\mathcal{D}'$, which is obtained from $\mathcal{D}$ *in a particular way* (i.e., by performing a particular reduction step) be computable. This yields the notion defined by Martin-Löf (1971). One might even weaken this by not referring to a particular procedure and just postulate in (iii) that $\mathcal{D}$ reduces to a computable $\mathcal{D}'$, without specifying the procedure in the definition (it has then to be specified in the normalization proof, of course).

II.1.2 From computability to validity

Validity is a core notion of proof-theoretic semantics. Prawitz introduced it as a semantical predicate for derivations, in analogy to truth as a semantical predicate of propositions in model-theoretic semantics. He developed it in connection with computability predicates, to which it bears a strong resemblance. As his terminology ("strong validity" for "computability" in our sense) suggests, Prawitz actually considers computability and validity to be concepts on one scale, computability being the stronger one. There are several remarks in his 1971, 1973, 1974 papers, where he deals

with both notions, which indicate that computability is obtained by augmenting validity, some of them even saying that these extensions modify validity in a way which makes it even more plausible or convenient.[14] However, Prawitz never states the exact relationship between both concepts. In particular, he never attempts to formally prove that computability (strong validity) implies validity — a result one should expect to hold if the relationship is as simple as the terminology suggests. In his publications after 1974 Prawitz never comes back to computability and its relation to validity.

In the following I shall argue that, in spite of many similarities, and contrary to Prawitz's opinion, semantically useful validity notions have to differ considerably from computability. Crucial modifications are necessary to turn computability into validity. I shall make the following points:

(1) The notion of computability is not suitable as a foundational semantic notion of the validity of derivations, because it stipulates normal derivations as computable without further justification.

(2) In order to adjust the notion of computability to serve for foundational purposes, *closed* derivations have to be given a distinguished role in the justification of irreducible (= normal) derivations.

(3) This distinguished role of closed derivations includes, as a semantical condition, their reducibility to canonical form.

Ad (1): Computability is not a semantical notion

According to clause (ii) in the definition of computability, every normal derivation which is not in I-form, is computable[15]. This could be counted as a semantical clause only if in proof-theoretic semantics we are prepared to consider non-canonical normal derivations as valid *by definition*. However, as we have seen, it is one of the ideas of proof-theoretic semantics in the sense of Gentzen's programme to consider introduction inferences as basic and to *justify* all other inferences from them. In other words, only derivations based on introduction rules should be taken for granted. In any other case the definition of validity should rely on some *justification* procedure rather than on the syntactic form of derivations. This is obviously violated by clause (ii) which just stipulates irreducible non-canonical derivations as valid. There is no semantical reason whatsoever to consider non-canonical irreducibility as a definition case of validity. According to such a definition, the derivation

$$\frac{A{\to}B \quad A}{B}$$

[14]See Prawitz (1971), p. 289; Prawitz (1973), p. 238.

[15]In other renderings of computability, all normal derivations are computable by definition, not only those which are not in I-form. For the definition of computability chosen here this follows as a lemma.

would be valid *by definition* and not *by justification*, which is not what is wanted. Modus ponens, as an elimination rule, definitely needs semantical justification. Of course, for the purpose of proving normalizability, clause (ii) is absolutely natural, as normal derivation are trivially (strongly) normalizable. For semantical purposes, however, we would have to argue that non-canonical irreducible derivations have some special status, which exempts them from justification. Since for that no argument is at hand, using normal derivations as a starting point in defining validity is an ill-guided approach.

In contradistinction to clause (ii), clauses (i) and (iii) make good semantic sense. In terms of validity, clause (i) says that a canonical derivation of $A \to B$ is valid if its immediate predecessor, a derivation of B from A provides a way of transferring every valid derivation of A into a valid derivation of B, which corresponds to the meaning one wants to associate with $A \to B$. Furthermore, clause (iii) says that a non-canonical derivation may be considered as valid if it reduces to a valid derivation. This reflects the idea that non-canonical derivations are valid if they reduce to derivations which are already justified as valid (such as canonical ones).

Therefore the *basic flaw* in computability, understood as a semantical notion, is the following implicit assumption:

If $\mathcal{D}$ is non-canonical and irreducible (= normal), then $\mathcal{D}$ is valid.

Ad (2): Semantically modified computability: open assumptions and closed derivations

One could try to modify the definition of computability to make it suitable for a definition of semantical validity. This would mean that clause (ii) of the definition is dropped and replaced with something which justifies non-canonical irreducible derivations as valid. An obvious possibility would be to consider such a derivation as valid if replacement of assumption formulas with valid derivations yields a valid derivation of the end formula. This idea would follow the substitution lemma for computability, according to which computability is the same as computability under substitution. More formally, clause (ii) would then read as follows (where we now use the term "valid", as we are dealing with turning the computability notion into a semantical concept):

(ii) A non-canonical irreducible derivation*

$$\begin{array}{l} A_1 \dots A_n \\ \quad \mathcal{D} \\ \quad B \end{array}$$

where all open assumptions of $\mathcal{D}$ are among $A_1, \dots, A_n$, is valid, if for every list of

valid derivations $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$, $\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ B\end{matrix}$ *is valid.*

For example, the one step non-canonical irreducible derivation

$$\frac{A{\to}B \quad A}{B}$$

would be considered as valid, if for each pair of valid derivations $\begin{matrix}\mathcal{D}_1\\ A{\to}B\end{matrix}$ and $\begin{matrix}\mathcal{D}_2\\ A\end{matrix}$, the derivation $\dfrac{\begin{matrix}\mathcal{D}_1 & \mathcal{D}_2\\ A{\to}B & A\end{matrix}}{B}$ is valid. However, a clause like (ii)* would then no longer proceed by induction on the complexity of the end formula but on the complexity of the assumption formulas plus that of the end formula, in the example: on the complexities of $A{\to}B, A$ and B. But then the quantification over all valid derivations of the assumption formulas is no longer feasible, since they themselves may have assumptions of any complexity. Therefore this is no viable solution.[16]

The way out used in semantical definitions of validity is to use *closed* valid proofs rather than arbitrary valid proofs as a basis. Instead of (ii)* one would then propose the following clause.

*(ii)** A non-canonical non-reducible derivation*

$$\begin{matrix}A_1 \ldots A_n\\ \mathcal{D}\\ B\end{matrix}$$

where all open assumptions of $\mathcal{D}$ *are among* $A_1, \ldots, A_n$*, is valid, if for every list of* closed *valid derivations* $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$,

$$\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ B\end{matrix} \quad \textit{is valid.}$$

[16] It is bound to fail due to the impredicative character of the substitution lemma, when it is turned into a definition. "Impredicative" here means that computability is defined by quantifying over all substitution instances obtained by substituting arbitrary computable derivations.

However, even now we are proceeding by induction over the joint complexity of $A_1, \ldots, A_n, B$ rather than only the complexity of B, even if we only quantify over *closed* valid derivations. This is not compatible with clause (i) where we proceed by induction over the end formula only. In order to cope with that, we would also have to change clause (i) to

*(i)** A derivation of the form* $\dfrac{\begin{array}{c}[A]\\ \mathcal{D}\\ B\end{array}}{A\to B}$ *is valid, if for every* closed *valid* $\begin{array}{c}\mathcal{D}'\\ A\end{array}$, $\begin{array}{c}\mathcal{D}'\\ A\\ \mathcal{D}\\ B\end{array}$ *is valid*

where this is understood as proceeding by induction over the joint complexity of open assumptions plus end-formula of a derivation.

The definition based on (i)**, (ii)** and (iii) may be called *validity**. What we have done in passing from computability to validity* is *interpreting open assumptions as placeholders for closed derivations.*

Ad (3): The reducibility of closed derivations

Unfortunately, validity* does not yet eliminate the possibility that irreducible (= normal) derivations are counted as valid without any further justification. In the case of open derivations this possibility has been removed, but not so in the case of closed derivations. Suppose $\mathcal{D}$ is a closed non-canonical derivation which is irreducible. Then clause (ii)** applies, and, as there are no open assumptions, $\mathcal{D}$ is (vacuously) valid*.

One might argue that there are no closed non-canonical irreducible derivations anyway. However, this is just an accidental property of first-order logic with the standard reductions. Since the notion of validity should in principle be applicable to more general notions of derivations and reductions, the formal possibility of closed non-canonical irreducible derivations should be taken into account. They should simply turn out as invalid by definition (rather than as valid in case of validity*). This is accomplished by turning a corrollary of normalization of proofs into a semantical condition:

A closed non-canonical derivation is valid, if it is reducible to a valid closed canonical derivation.

It was Dummett in particular who stressed at many places as a fundamental epistemological principle that, if something is known in an indirect (non-canonical) way, it must be possible to turn this indirect knowledge into direct (canonical) knowledge. This is part of the reason why this sort of semantics is also called verificationist, and it is part of the interpretation of Gentzen's programme of the primacy of introduction rules: In the closed case an I-rule derivation can always be found. With this motivation we arrive at Prawitz's definition of the validity of derivations.

II.1.3 Validity of derivations

We follow Prawitz (1971) in defining validity with respect to atomic systems S which are given by production rules for atomic formulas. Let then $\mathcal{L}(S)$ be implicational logic over S, i.e. the system given by introduction and elimination rules for implication plus the production rules of S. We may identify $\mathcal{L}(S)$ with the set of all derivations in this system. A system S' is an *extension* of S ($S' \geq S$) if S' is S itself or results from S by adding further production rules. As a limiting case we consider the empty atomic system S_0 without any inference rule and with propositional variables as formulas, and correspondingly $\mathcal{L}(S_0)$ as standard implicational logic over propositional variables. Obviously, as a formal system, $\mathcal{L}(S_0)$ is the same as $\mathcal{L}$. It will turn out that validity with respect to S_0 is the same as *universal* validity when defined in an appropriate way. We say that $\mathcal{D}$ *reduces to* $\mathcal{D}'$ ($\mathcal{D} \succeq \mathcal{D}'$), if $\mathcal{D}'$ can be obtained from $\mathcal{D}$ by applying a (finite) number of reduction steps. As a limiting case, $\mathcal{D}$ reduces to itself. In the context of atomic systems, we also extend the notion of a canonical derivation. A canonical derivation of an atom of S is a derivation in S, whereas, as before, a canonical derivation of a complex formula is a derivation in I-form, i.e., a derivation using an introduction rule in the last step.

Then our first definition of validity corresponding to the one given in Prawitz (1971) runs as follows:

Definition of S-validity 1
(i) For atomic A, a closed derivation of A is S-valid, if it reduces to a derivation in S.

(ii) A closed derivation $\begin{matrix}\mathcal{D}\\ A\to B\end{matrix}$ *is S-valid, if $\mathcal{D}$ reduces to a derivation of the form*

$$\dfrac{\begin{matrix}[A]\\ \mathcal{D}'\\ B\end{matrix}}{A\to B}$$

such that for every $S' \geq S$ and every closed S'-valid $\begin{matrix}\mathcal{D}''\\ A\end{matrix}$, $\begin{matrix}\mathcal{D}''\\ A\\ \mathcal{D}'\\ B\end{matrix}$ *is S'-valid.*

(iii) An open derivation $\begin{matrix}A_1 \ldots A_n\\ \mathcal{D}\\ B\end{matrix}$, *where all open assumptions of $\mathcal{D}$ are among $A_1, \ldots, A_n$, is S-valid, if for every $S' \geq S$ and every list of closed S'-valid* $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$, $\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ B\end{matrix}$ *is S'-valid.*

This inductive definition proceeds over the joint complexity of open assumptions and end formula of the given derivation.

In view of clause (iii), clause (ii) can be changed to

(ii) A closed derivation of $A \to B$ is S-valid if it reduces to a canonical derivation of $A \to B$ whose immediate subderivation is S-valid.

By putting reduction into a clause of its own, the whole definition can then be equivalently stated as follows.

Definition of S-validity 2

(I) Every closed derivation in S is S-valid.

(II) A closed canonical derivation of $A \to B$ is S-valid, if its immediate subderivation is S-valid.

(III) A closed non-canonical derivation is S-valid, if it reduces to an S-valid canonical derivation.

(IV) An open derivation $\begin{array}{c} A_1 \dots A_n \\ \mathcal{D} \\ B \end{array}$ *, where all open assumptions of $\mathcal{D}$ are among $A_1, \dots, A_n$, is S-valid, if for every $S' \geq S$ and for every list of closed S'-valid* $\begin{array}{c} \mathcal{D}_i \\ A_i \end{array}$

$(1 \leq i \leq n)$, $\begin{array}{c} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \dots A_n \\ \mathcal{D} \\ B \end{array}$ *is S'-valid.*

The equivalence of these two definitions of S-validity is easy to prove. Obviously, every (not necessarily closed) derivation in S is S-valid, since every closed S-valid derivation of an atom reduces to a derivation in S.

This definition corresponds to the one proposed in Prawitz (1974) and also in his contribution to this volume. As explained in the last subsection, the philosophical motivation behind this definition is that, in the closed case, derivations in S as well as introduction steps are self-justifying (clauses I and II), whereas all other steps are justified on the basis that they *reduce* to something which is already justified (clause III), or, in the open case, produce justified closed derivations when combined with such ones (clause IV).

The reason for considering arbitrary extensions S' of S is to block arguments for S-validity based on the underivability of certain formulas in S. Otherwise, for example, every derivation in $\mathcal{L}$ starting with a propositional variable as an open assumption should be counted as S_0-valid, because there is no closed derivation of a propositional

variable in S_0. In this sense the consideration of extensions $S' \geq S$ is a *monotonicity* condition for S-validity. S-valid derivations should remain S-valid if one's knowledge incorporated in the atomic system S is increased.[17] In fact, it is easy to show that we have a

Monotonicity theorem for S-validity
A derivation $\mathcal{D}$ in $\mathcal{L}(S)$ is S-valid iff for every $S' \geq S$, $\mathcal{D}$ is S'-valid.

Investigating the consequences of permitting non-monotonicity of S-validity is beyond the scope of this paper.

As compared to computability, this definition relies on two crucial insights:

(1) The distinction between closed and open derivations is primary as compared to that between canonical and non-canonical derivations. The latter plays its role as a subdistinction within closed derivations. In the definition of S-validity we proceed according to the concept tree

$$\begin{vmatrix} closed \begin{vmatrix} canonical \\ non-canonical \end{vmatrix} \\ open \end{vmatrix}$$

whereas the definition of computability rests on

$$\begin{vmatrix} canonical \\ non-canonical \begin{vmatrix} reducible \\ irreducible \end{vmatrix} \end{vmatrix}$$

In S-validity, the closed/open distinction is the fundamental one, since closed canonical derivations carry the burden of semantical justification and are self-justifying. In computability, irreducible (= normal) derivations form the basis, i.e., non-canonical irreducible derivations are self-justifying.[18]

(2) The reduction clause for closed derivations (clause (III)) uses an existence condition corresponding to *weak* normalization, which is again due to the self-justifying character of closed canonical erivations. Whereas in computability, self-justifying derivations are by definition tied to the reducibility concept, viz. as derivations which are *irreducible*, in S-validity self-justifying derivations are defined independently of reducibility and are not trivially available when a derivation is not reducible, which means that we have to postulate their existence as a result of reduction.

---

[17] I suppose that Prawitz had something similar in mind (see Prawitz 1971, p. 276). In later papers he drops considering extensions $S' \geq S$, and only considers extensions of justifying procedures (see below section II.3).

[18] This does not mean that S-validity of closed and of open derivations is defined separately. These two cases occur intertwined in the same derivation. This is due to the fact that the immediate subderivation of a closed canonical derivation of $A \to B$ is a derivation of B *from* the assumption A.

For our case of implicational logic we can easily show the following:

Soundness theorem for S-validity *For any S, every derivation in $\mathcal{L}(S)$ is S-valid.*

II.1.4 Validity and universal validity

Universal validity will be defined for derivations in $\mathcal{L}$. Intuitively, a derivation in $\mathcal{L}$ should be universally valid if it is S-valid for every S. For that we have to interpret derivation of $\mathcal{L}$ in $\mathcal{L}(S)$. Let an S-assignment v be a mapping of propositional variables to S-formulas. Then for an $\mathcal{L}$-derivation $\mathcal{D}$, $\mathcal{D}^v$ is the $\mathcal{L}(S)$-derivation resulting from $\mathcal{D}$ by replacing every propositional variable with the corresponding S-formula assigned to it via v. We can then say that $\mathcal{D}$ is *valid in S under v*, if $\mathcal{D}^v$ is S-valid in the sense defined in the previous section. $\mathcal{D}$ is then called valid in S if it is valid in S under every v, and it is called *universally valid*, if it is valid in S for every S. Now the following holds.

Proposition *Let $\mathcal{D}$ be a derivation in $\mathcal{L}$. Then $\mathcal{D}$ is universally valid iff $\mathcal{D}$ is S_0-valid.*

Proof One has to use that, when $\mathcal{L}$ is interpreted in $\mathcal{L}(S)$, every extension $S' \geq S$ can be viewed as an interpretation of an extension of S_0 via an assignment.

Therefore, from now on we use the term "valid" terminologically as meaning universal or S_0-validity.

Then as a corrollary of the soundness theorem for S-validity we have the following:

Soundness theorem for validity *Every derivation in $\mathcal{L}$ is valid.*

As we have a corresponding theorem for computability (Proposition 2), and as we are so far only taking derivations in implicational logic into consideration, computability and validity coincide in the sense that any computable derivation (i.e., any derivation in implicational logic) is a valid derivation (i.e., a derivation in implicational logic) and vice versa. So extensionally, computability and validity coincide. We can differentiate between them extensionally when we consider more general notions of derivation structures. Then we can give actual counterexamples which show that computability and validity differ not only with respect to their contents but are in fact *extensionally* different concepts (see below section II.4). This further substantiates our claim that, contrary to Prawitz, computability is at best a forerunner to validity but not a semantical concept itself.

II.1.5 Validity concepts which imply normalizability: strict and strong validity

Our basic semantical argument against computability and for validity was that irreducible derivations should never be counted as valid without further justification, i.e. the implication

irreducible implies valid

should *not* hold by definition. One might, however, expect that

valid implies normalizable

holds[19]. According to the present definition of validity, normalizability is not implied by validity. If we consider intuitionistic logic with no introduction rule for absurdity $\perp$, then according to our definition of validity, $\dfrac{\perp}{\mathcal{D}}$ is vacuously valid for any $\mathcal{D}$ with $\perp$ as the only open assumption, even if $\mathcal{D}$ is not normalizable. Now one might argue that a semantical justification of open derivations in terms of substitution with closed valid derivations should only be applied if the derivation is reduced as far as possible, and not already in the present case, where $\mathcal{D}$ can still be reduced. This means that the substitution justification in clause (IV) of the definition of S-validity should be put into action only if all possibilities of obtaining a justification by means of reduction are exhausted, i.e., when the derivation in question is irreducible. Calling this notion "strict S-validity" (or "strict validity" [simpliciter] for the universal concept), we reach the following definition:

Definition of strict S-validity
(I) Every closed derivation in S is strictly S-valid.
(II) A closed canonical derivation of $A \to B$ is strictly S-valid, if its immediate subderivation is strictly S-valid.
(III) A closed non-canonical derivation is strictly S-valid, if it reduces to a strictly S-valid canonical derivation.
(IV) An open reducible derivation is strictly S-valid, if it reduces to a strictly S-valid derivation.
(V) An open irreducible derivation $\begin{matrix} A_1 \dots A_n \\ \mathcal{D} \\ B \end{matrix}$ *, where all open assumptions of $\mathcal{D}$ are among $A_1, \dots, A_n$, is strictly S-valid, if for every $S' \geq S$ and for every list of closed*

[19]This is not exactly the converse, which would be *"valid implies normal"*, which is, of course, wrong.

and strictly S'-valid $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$, $\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ B\end{matrix}$ *is strictly S'-valid.*

The difference to the definition of S-validity consists in the fact that clause (IV) is split up into clauses (IV) and (V), where the new clause (IV) asks for the reduction of reducible open derivations, while the new clause (V) is the old clause (IV), but applied only to the irreducible case. So the conceptual tree of this definition is the following one

$$\left|\begin{matrix}closed\left|\begin{matrix}canonical\\ non-canonical\end{matrix}\right.\\ open\left|\begin{matrix}reducible\\ irreducible\end{matrix}\right.\end{matrix}\right.$$

which constrasts sharply with computability, where the reducible/irreducible distinction is a subdistinction of non-canonical derivations.

I speak of "strict" rather than "strong" S-validity to distinguish it from Prawitz's notion of strong validity, which corresponds to computability, and from associations with strong normalization. Furthermore, I want to reserve "strong S-validity" for a notion defined below for which this association is justified. Strict S-validity as considered here is indeed a notion on the same scale as S-validity. It is obvious that strict S-validity implies S-validity, but not necessarily vice versa.[20] The corresponding universal notion of strict validity (simpliciter) is defined as in the previous section (II.1.4).

Let us define (weak) normalizability inductively as follows.

Definition of normalizability
(i) Every canonical derivation is normalizable if its immediate subderivation is normalizable.
(ii) Every non-canonical normal derivation is normalizable.
(iii) Every non-canonical reducible derivation is normalizable, if it reduces to a normalizable derivation.

Then we can formulate as a theorem that strict validity implies (weak) normalizability.

Theorem *Every strictly valid derivation is normalizable.*

[20] Again some emphasis has to be put on "necessarily", as in the case of intuitionistic logic, all derivations are strictly S-valid, i.e., strict S-validity and S validity coincide in this case.

By *strong S-validity* we denote an even more strenghtened concept, which implies strong normalization. However, the concept defined is still different from computability, as we not just require outright that every reduction sequence terminates. Rather, this is only required in the reducible case, in order not to render any normal derivation strongly valid.

Definition of strong S-validity
(I) Every closed derivation in S is strongly S-valid.
(II) A closed canonical derivation of $A \to B$ is strongly S-valid, if its immediate subderivation is strongly S-valid.
(III) A closed non-canonical derivation $\mathcal{D}$ is strongly S-valid, if $\mathcal{D}$ is reducible and if for every $\mathcal{D}'$ such that $\mathcal{D} \succ_1 \mathcal{D}'$, $\mathcal{D}'$ is strongly S-valid.
(IV) An open reducible derivation $\mathcal{D}$ is strongly S-valid, if for every $\mathcal{D}'$ such that $\mathcal{D} \succ_1 \mathcal{D}'$, $\mathcal{D}'$ is strongly S-valid.

(V) An open irreducible derivation $\begin{matrix} A_1 \dots A_n \\ \mathcal{D} \\ B \end{matrix}$*, where all open assumptions of $\mathcal{D}$ are among $A_1, \dots, A_n$, is strongly S-valid, if for every $S' \geq S$ and for every list of closed and strongly S'-valid* $\begin{matrix} \mathcal{D}_i \\ A_i \end{matrix}$ $(1 \leq i \leq n)$, $\begin{matrix} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \dots A_n \\ \mathcal{D} \\ B \end{matrix}$ *is strongly S'-valid.*

Obviously, strong S-validity implies strict S-validity.

A corresponding universal notion of strong validity (simpliciter) is defined as in the previous section (II.1.4).

We extend the definition of normalizability to a definition of strong normalizability by replacing, in clause (iii) of this definition, "if it reduces to" with "if every derivation it reduces to in a single step is". In analogy with the case of strict validity we can show that strong validity implies strong normalizability.

Theorem *Every strongly valid derivation is strongly normalizable.*

There are also soundness theorems for strict and strong [S-]validity.

Soundness theorems for strict and strong [S-]validity
All [S-]derivations are both strictly and strongly [S-]valid.

With strict and strong validity we have obtained concepts, which are semantically satisfying and at the same imply weak and strong normalization, respectively. Compared to them, computability is a starting point rather than a relevant concept in the final semantic setting.

II.2 Validity and computability based on elimination rules

It is a central idea of proof-theoretic semantics to consider one set of rules as basic and justify derivations based on other rules with respect to them as valid. The standard approach is to take the introduction rules as primitive or "self-justifying" (Dummett). However, as envisaged by Prawitz[21], one might try to take the opposite way and start with elimination inferences. Prawitz's presentation is very sketchy. I reconstruct it as follows.

According to the I-rule conception, if in $\begin{matrix}\mathcal{D}\\A\end{matrix}$ the formula A is the conclusion of an introduction rule whose premiss derivation is S-valid, then $\begin{matrix}\mathcal{D}\\A\end{matrix}$ is S-valid by definition. If $\begin{matrix}\mathcal{D}\\A\end{matrix}$ is not derived by an introduction rule, it is S-valid if it can be *reduced* to an S-valid derivation. Analogously, one might postulate within an E-rule conception that, if all applications of elimination rules to $\begin{matrix}\mathcal{D}\\A\end{matrix}$ yield S-valid derivations, then $\begin{matrix}\mathcal{D}\\A\end{matrix}$ is itself S-valid by definition. If no elimination rule can be applied to $\begin{matrix}\mathcal{D}\\A\end{matrix}$, then it is S-valid if it can be *reduced* to an S-valid derivation. (Obviously, the latter case only arises when A is atomic.)

This way of reasoning suggests the following definition (again only for implicational logic).

Definition of S-validity based on E-rules
(I) Every closed derivation in S is S-valid$_E$.

(II) A closed derivation $\begin{matrix}\mathcal{D}\\A\to B\end{matrix}$ *of* $A\to B$ *is S-valid$_E$, if for every $S' \geq S$ and every closed S'-valid$_E$* $\begin{matrix}\mathcal{D}'\\A\end{matrix}$, *the (closed) derivation* $\dfrac{\begin{matrix}\mathcal{D}\\A\to B\end{matrix}\quad\begin{matrix}\mathcal{D}'\\A\end{matrix}}{B}$ *is S'-valid$_E$.*

(III) A closed derivation $\begin{matrix}\mathcal{D}\\A\end{matrix}$ *of an atomic formula A, which is not a derivation in S, is S-valid$_E$, if it reduces to a derivation in S.*

(IV) An open derivation $\begin{matrix}A_1 \ldots A_n\\\mathcal{D}\\B\end{matrix}$, *where all open assumptions of $\mathcal{D}$ are among*

[21] Prawitz (1971), p. 289seq. [= appendix A.2]

$A_1, \ldots, A_n$, *is* S-*valid*$_E$, *if for every* $S' \geq S$ *and every closed* S'-*valid*$_E$ $\begin{matrix}\mathcal{D}_i \\ A_i\end{matrix}$ $(1 \leq i \leq n)$,

$$\begin{matrix}\mathcal{D}_1 & & \mathcal{D}_n \\ A_1 & \ldots & A_n \\ & \mathcal{D} & \\ & B & \end{matrix}$$ *is* S'-*valid*$_E$.

Clause (IV) is identical with clause (IV) in the definitions of S-validity in section II.1.3, i.e., open assumptions in derivations are interpreted in the same way as before. Clauses (I) and (III) can be conjoined to the single clause

(I/III) A closed derivation $\begin{matrix}\mathcal{D} \\ A\end{matrix}$ *of an atomic formula* A *is* S-*valid*$_E$, *if it reduces to a derivation in* S.

Using the main reductions, it can again be shown that all derivations in $\mathcal{L}(S)$ are S-valid$_E$. As Prawitz remarks, this approach only works for logical constants with "direct" elimination rules such as $\rightarrow$ $\wedge$ and $\forall$. There is no way to carry it over to constants like $\vee$ and $\exists$ with "indirect" elimination rules.

Corresponding to the procedure in section II.1.5, notions of strict S-validity$_E$ and strong S-validity$_E$ can be defined such that strict S-validity$_E$ implies weak normalizability and strong S-validity$_E$ implies strong normalizability.[22]

There is also a corresponding notion of computability based on elimination rules for the purpose of strong normalization proofs. Actually, it is more common in today's presentations than computability based on introduction rules, as long as one does not deal with $\exists$ or $\vee$. For example, Troelstra & Schwichtenberg (1996) define computability as follows:

Definition of computability based on E-rules

(1) For atomic A, $\begin{matrix}\mathcal{D} \\ A\end{matrix}$ *is computable*$_E$, *if* $\begin{matrix}\mathcal{D} \\ A\end{matrix}$ *is strongly normalizable.*

[22] However, in the case of strict S-validity$_E$ (not in the case of strong S-validity$_E$) we would have to distinguish between reducible and irreducible derivations not only in the open case, but also in the closed case, i.e., clause (II) should only be applicable if $\begin{matrix}\mathcal{D} \\ A \rightarrow B\end{matrix}$ has been reduced as far as possible, which means that it is irreducible. Otherwise we cannot prove that $\begin{matrix}\mathcal{D} \\ A \rightarrow B\end{matrix}$ is weakly normalizable given that $\dfrac{\begin{matrix}\mathcal{D} & \mathcal{D}' \\ A \rightarrow B & A\end{matrix}}{B}$ is weakly normalizable (in the case of strong normalizability this is trivial).

(2) $\begin{matrix}\mathcal{D}\\ A\end{matrix}$ *is computable*$_E$*, if for every computable*$_E$ $\begin{matrix}\mathcal{D}'\\ A\end{matrix}$, $\dfrac{\begin{matrix}\mathcal{D}\\ A\to B\end{matrix}\quad\begin{matrix}\mathcal{D}'\\ A\end{matrix}}{B}$ *is computable*$_E$.

Similar to computability based on I-rules this notion has again the feature that normal derivations — here even normalizable ones — are counted as computable$_E$ without further justification, which is natural for proving normalization but cannot be used for a semantical concept.

As a characteristic feature of the defintions of validity$_E$ and computability$_E$ based on E-rules, it might be noted that the notion of reduction only comes in at the atomic case (in the definition of computability$_E$ in the form of a derivation being strongly normalizable). In the terminology of terms one might say that everything is played down to the atomic level by means of term application, whereas the I-rule conceptions were based on what corresponds to term substitution.

The conception sketched here is not the only possible and perhaps not even the most genuine way of putting elimination rules first. If one really tried to dualize the I-rules approach and to put "deriving from" rather than "deriving of" in front, one should develop ideas like the following. A *closed derivation from* A should be a derivation of absurdity from A, and a derivation $\begin{matrix}A\\ \mathcal{D}\\ B\end{matrix}$ should be justified, if, for every closed valid derivation $\begin{matrix}B\\ \mathcal{D}'\end{matrix}$ from B, $\begin{matrix}A\\ \mathcal{D}\\ B\\ \mathcal{D}'\end{matrix}$ is a closed valid derivation from A, etc. This, however, produces conflicts with the asymmetry of derivations, which normally have exactly one end formula, but possibly several assumption formulas. Full dualization would perhaps lead to a sort of single-premiss or multiple-conclusion logic. Some sort of genuine E-rule approach might be desirable, if one wanted to logically elaborate ideas like Popper's falsificationism by putting refutation at the basis of reasoning.[23]

II.3 Derivation structures, justifications and arguments

The soundness theorems for derivations in $\mathcal{L}$ are interesting metalogical facts. However, of a semantical notion of validity we expect more than that. Validity should be a *distinguishing* feature, telling that some derivations are valid and others are not. This is quite analogous to the notion of truth which tells that some propositions are true

[23]Dummett (1991, Ch. 13, p. 283-286) attempts to develop some kind of a "genuine" E-rule approach (within the standard setting of derivations with multiple premisses and single conclusions).

whereas others are not true. A result showing that every proposition is true, making truth a general feature of propositions, would be considered inadequate. Similarly, there should be some more general notion of derivation among which the notion of validity determines a *subclass*. It is easy to construct such derivations: just combine arbitrary rules, not just the rules which belong to $\mathcal{L}$. For example, a single step derivation in $\mathcal{L}$ of the form

$$\frac{A \to B}{B}$$

should turn out as not being valid, because for certain $S \geq S_0$, not every closed S-valid derivation of $A \to B$ becomes a closed S-valid derivation of B when B is appended at the end of this derivation.[24] This means that we have to be able to talk about arbitrary derivations which are not built according to a previously given set of rules. This is particularly required if one wants to pose the question of *completeness*, i.e. the question of whether every valid derivation can be represented in $\mathcal{L}$. As long as [S-]validity is only defined for $\mathcal{L}$ or $\mathcal{L}(S)$, completeness is absolutely trivial. It just says that every [S-]valid derivation is a derivation, as there are no candidates for derivations which might not be in $\mathcal{L}$ or $\mathcal{L}(S)$.

Since by "derivations" one normally understands derivations in a given system, one should choose a different term. I propose to talk about *derivation structures*. So the purpose of this section is to define a notion of a derivation structure and of the [S-]validity of derivation structures. Derivations in $\mathcal{L}$ or $\mathcal{L}(S)$ are then just derivation structures generated by particular rules of inference.

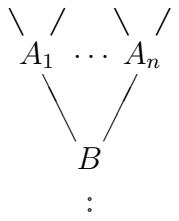
This problem does not arise when we are dealing with computability and normalizability only. Computability, as an auxiliary concept to prove normalization, is not necessarily a concept which aims at dividing derivations in computable and non-computable ones, at least not in the first place. In the context of computability, we just want to show *all* derivations have the property of being normalizable.

In order to develop a notion of derivation in a generalized sense, we take up concepts from the theory of natural deduction and carry them over to arbitrary formula trees. A derivation structure over the language of implicational logic (and possibly over atomic systems S as well) can be defined as follows. A *derivation structure* is a *formula tree* together with a *discharge function*. A discharge function for a formula tree is a function which associates with every top formula[25] a formula occurring below (on the

[24]For example, if A and B are propositional variables, we may choose S as having no axiom and $A \Rightarrow B$ as the only inference rule. Then there is an S-valid derivation of $A \to B$, but no S-valid derivation of B.

[25]More precisely, we should talk of top formula *occurrences*. I do not always terminologically distinguish between formulas and their occurrences. It will always be clear from the context what is meant.

same branch in the tree). The intended reading is the following. Suppose $A_1, \ldots, A_n$ and B occur in the tree as follows:



where each A_i is the value of the discharge function f for top-formulas $C_{i1}, \ldots, C_{im_i}$, i.e., $f(C_{i1}) = \ldots = f(C_{im_i}) = A_i$. Then B is inferred as a conclusion from the premisses $A_1, \ldots, A_n$, where at this application, for each i $(1 \leq i \leq n)$, the assumptions $C_{i1}, \ldots, C_{im_i}$ in the derivation of each A_i are discharged. This means that the derivation represents the step governed by the following inference rule:

$$\frac{\begin{array}{cccc}[C_{11}, \ldots, C_{1m_1}] & & [C_{n1}, \ldots, C_{nm_n}] & \\ A_1 & \ldots & A_n & \end{array}}{B}\ (R)\ .$$

In this way, inference rules can be extracted from a derivation structure, and it can be checked if a given set of inference rules allows one to generate this derivation structure.[26] It should be noted, however, that the concept of a derivation structure is independent of the concept of an inference rule. This neatly fits with the notion of validity, which refers to derivations rather than to rules.

According to this definition,

$$\frac{A \to B}{B}$$

is a (very simple) derivation structure. Therefore, once we have defined S-validity and validity for derivation structures, we are in the position to formulate that this derivation structure is not valid. We may then also pose the question of whether every valid derivation structure is or can be represented as a derivation in $\mathcal{L}$, which corresponds to *completeness*. Actually, completeness can be formulated in our context in two ways:

(Compl 1) Every valid derivation structure is a derivation in $\mathcal{L}$.

(Compl 2) If there is a valid derivation structure with end formula A and open assumptions $B_1, \ldots, B_n$, then there is a derivation of A in $\mathcal{L}$, all of whose open assumptions are among $B_1, \ldots, B_n$.

[26] Such a notion of a derivation structure is spelled out in detail in Schroeder-Heister 1984a,b.

It is obvious that *(Compl 2)* follows from *(Compl 1)*, but not necessarily vice versa. *(Compl 1)* says that every valid derivation structure *is itself* a derivation in $\mathcal{L}$, whereas *(Compl 2)* only says that for every valid derivation structure a derivation in $\mathcal{L}$ (with the same end formula and no additional assumptions) *can be found*.

It is easy to prove completeness in the sense of *(Compl 1)* with respect to the present notion of validity, where we replace "derivation" with "derivation structure" and understand reductions as before. Proceeding by induction on the complexity of valid derivation structures we argue as follows: A derivation structure $\langle A, f\rangle$ with $f(A) = A$ is a derivation in $\mathcal{L}$, which just consists of the assumption formula A. Suppose a derivation structure $\mathcal{D}$ with end formula B has been constructed using a rule (R). If (R) is a primitive rule of $\mathcal{L}$, we just apply the induction hypothesis. Suppose (R) is not a primitive rule of $\mathcal{L}$. Suppose furthermore that $\mathcal{D}$ is closed. Then $\mathcal{D}$ is not canonical since (R) is not an introduction rule (and not an atomic rule either, as we can choose the atomic system without any rule). Then $\mathcal{D}$ is valid only if it reduces to a canonical derivation. This is impossible, however, since such a reduction can only affect the immediate subderivations of $\mathcal{D}$ as there is no reduction step involving (R). Therefore, $\mathcal{D}$ cannot be valid if (R) is not a primitive rule of $\mathcal{L}$. Since strong validity implies validity, this holds for strong validity as well.

Looking more closely at this completeness proof, it turns out to be quite trivial. The fact that valid derivation structures are derivations in $\mathcal{L}$ is due to the fact that reduction procedures are only associated with elimination rules. This means that if a closed derivation structure ends with a rule which is neither an introduction rules (which yields a valid derivation if the immediate subderivation is valid) nor with an elimination rule (which yields a valid derivation if it *reduces* to a derivation structure in I-form), it is not valid at all. Thus we enforce derivations in $\mathcal{L}$ to be the only valid ones, as they exhibit the pattern of I- and E-rules.

However, we would like to call a derivation structure valid if it uses steps that can be justified, even if they are not primitive steps of $\mathcal{L}$. For example, the one-step derivation

$$(\star)\ \frac{A\to(B\to C)}{B\to(A\to C)}$$

should be counted as valid, even if it is not a derivation in $\mathcal{L}$. For such a notion of validity, *(Compl 1)* would no longer hold, as $(\star)$ is not a primitive rule of $\mathcal{L}$. However, *(Compl 2)* might hold, and it would be striking if we could establish it as a result.

For such a revised notion of validity we would have to associate further reductions with derivation structures. E.g., if we added the reduction step

$$(\star\star)\qquad \dfrac{\begin{array}{c}\mathcal{D}\\ A\to(B\to C)\end{array}}{B\to(A\to C)} \quad \text{reduces to} \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\to(B\to C)\end{array}\qquad [A]\ (1)}{B\to C}\qquad [B]}{\dfrac{\dfrac{C}{A\to C}\,(1)}{B\to(A\to C)}\,(2)}$$

to our list of reduction steps, then the derivation $(\star)$ would indeed turn out as valid according to our definition of validity.

So what has actually to be changed is not so much the notion of validity itself, but the notion of reduction the definition of validity refers to. This fits very well with the general idea of reduction. Reductions serve as justifying procedures for non-canonical steps, i.e., for steps, which are not self-justifying. When we consider validity for arbitrary derivation structures, we should not just consider the topological structure of derivations but also generalize their reductions. This means that a more appropriate concept would be that of a derivation structure *together* with a set of permitted reduction steps, which need not coincide with the set of standard reductions used in normalization of derivations in $\mathcal{L}$.

This is exactly the step taken by Prawitz in his (1973) and later publications, including his contribution to the present volume. As he wants to consider derivations as representations of arguments, he calls an *argument skeleton* what we are calling a *derivation structure*. An *argument* is then a pair $\langle \mathcal{D}, \mathcal{J} \rangle$ consisting of a derivation structure together with a *justification* $\mathcal{J}$, where a justification consists of a set of reduction rules for derivation structures. A reduction rule is a transformation of derivation structures whose value has the same end formula and no open assumptions beyond those in the argument (but possibly less). More precisely, as reductions are only required when a derivation structure is non-canonical, justifications are only defined for non-canonical derivation structures of the form

$$\dfrac{\mathcal{D}_1 \ \ldots \ \mathcal{D}_n}{A}\,(X)$$

where (X) is not an introduction inference or an inference in an atomic system S. This means that the transformations within a justification $\mathcal{J}$ are attached to (non-canonical) rules. It is furthermore required that with each rule not two different reductions are associated. This is expressed by saying that $\mathcal{J}$ is *consistent*. $\mathcal{J}'$ is called a *consistent extension* of $\mathcal{J}$ ($\mathcal{J}' \geq \mathcal{J}$) if $\mathcal{J}'$ is either $\mathcal{J}$ itself or results from $\mathcal{J}$ by adding reductions for rules for which there are no reductions in $\mathcal{J}$, and is furthermore consistent.

It is not necessarily required that the reductions be schematic in the sense that different instances of a rule schema require the same reduction. E.g., it is not excluded

that for *modus ponens*

$$\frac{A{\to}B \quad A}{B}$$

the reductions defined differ depending on what formulas A and B stand for. However, in normal circumstances, reductions will be given schematically, as it is the case with the standard reductions of intuitionistic logic. In any case, reductions within justifications have to be interchangeable with substitution, which in the propositional case means the following:

If a reduction j is defined for a derivation $\begin{matrix} A_1 \dots A_n \\ \mathcal{D} \end{matrix}$, such that

$$j\begin{pmatrix} A_1 \dots A_n \\ \mathcal{D} \end{pmatrix} = \begin{matrix} A_1 \dots A_n \\ \mathcal{D}' \end{matrix}, \text{ then for any } \begin{matrix} \mathcal{D}_1 \\ A_1 \end{matrix}, \dots, \begin{matrix} \mathcal{D}_n \\ A_n \end{matrix},$$

$$j\begin{pmatrix} \mathcal{D}_1 & & \mathcal{D}_n \\ A_1 & \dots & A_n \\ & \mathcal{D} & \end{pmatrix} = \begin{matrix} \mathcal{D}_1 & & \mathcal{D}_n \\ A_1 & \dots & A_n \\ & \mathcal{D}' & \end{matrix}.$$

In the quantified case interchange with substitution for individual variables has to be added. Therefore, with respect to the substitution of derivations or derivation structures for open assumptions, reductions are schematic indeed.

Let now $\mathcal{L}^*(S)$ be the *logic of arguments* over S, which may be identified with the set of arguments $\langle \mathcal{D}, \mathcal{J} \rangle$, where the derivation structure $\mathcal{D}$ is built up from implicational formulas over formulas of S as atoms, and $\mathcal{J}$ is a justification whose reductions are defined for such derivation structures. As a limiting case we again have $\mathcal{L}^*(S_0)$, in short $\mathcal{L}^*$, which uses only propositional variables as atoms. Standard implicational logic $\mathcal{L}$ would then be obtained by considering the set of all $\langle \mathcal{D}, \mathcal{J} \rangle$ such that $\mathcal{D}$ is a derivation in standard implicational logic, whereas $\mathcal{J}$ is fixed for all derivations and comprises exactly the standard reductions.

Then the S-validity of arguments $\langle \mathcal{D}, \mathcal{J} \rangle$, which is the same as the S-validity of derivation structures $\mathcal{D}$ with respect to consistent sets of justifications $\mathcal{J}$, is defined as follows.

Definition of S-validity for arguments

(I) Every closed derivation in S is S-valid with respect to every $\mathcal{J}$.

(II) A closed derivation structure $\dfrac{\begin{matrix} A \\ \mathcal{D} \\ B \end{matrix}}{A{\to}B}$ *is S-valid with respect to $\mathcal{J}$, if its immediate*

substructure $\begin{array}{c} A \\ \mathcal{D} \\ B \end{array}$ *is S-valid with respect to* $\mathcal{J}$.

(III) A closed non-canonical derivation structure is S-valid with respect to $\mathcal{J}$*, if it reduces with respect to* $\mathcal{J}$ *to a canonical derivation structure, which is S-valid with respect to* $\mathcal{J}$.

(IV) An open derivation structure $\begin{array}{c} A_1 \ldots A_n \\ \mathcal{D} \\ B \end{array}$*, where all open assumptions of* $\mathcal{D}$ *are among* $A_1, \ldots, A_n$*, is S-valid with respect to* $\mathcal{J}$*, if for every* $S' \geq S$ *and* $\mathcal{J}' \geq \mathcal{J}$*, and for every list of closed derivation structures* $\begin{array}{c} \mathcal{D}_i \\ A_i \end{array}$ $(1 \leq i \leq n)$*, which are* S'*-valid with respect to* $\mathcal{J}'$*,* $\begin{array}{c} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \ldots A_n \\ \mathcal{D} \\ B \end{array}$ *is* S'*-valid with respect to* $\mathcal{J}'$.[27]

The reason for considering, in clause (IV), besides extensions $S' \geq S$ of atomic systems, also consistent extensions $\mathcal{J}' \geq \mathcal{J}$ of justifications, is again, in my view, a monotonicity constraint. It is obvious that the following holds.

Monotonicity of S-validity (for arguments)
An argument $\langle \mathcal{D}, \mathcal{J} \rangle$ *in* $\mathcal{L}^*(S)$ *is S-valid iff for every* $S' \geq S$ *and* $\mathcal{J}' \geq \mathcal{J}$*,* $\langle \mathcal{D}, \mathcal{J}' \rangle$ *is* S'*-valid.*

It should be noted that the definition of S-validity for arguments covers the case of S-validity of derivations in the old sense as a limiting case: Since for standard derivations, every inference rule which is not an introduction rule has already been assigned a reduction, there is no proper consistent extension of the standard set of reductions in this case. This means that the consideration of consistent extensions in the general case does not add anything specific to the standard case.

The corresponding universal concept is then defined as follows: If v is an assignment of S-formulas to propositional variables, then for a $\mathcal{J}$ comprising reductions for arguments in $\mathcal{L}^*$, $\mathcal{J}^v$ is defined to be the set of reductions which acts on derivations $\mathcal{D}^v$ in the same way as $\mathcal{J}$ acts on $\mathcal{D}$ (i.e., $\mathcal{J}^v$ is the homomorphic image of $\mathcal{J}$ under v). Then an argument $\langle \mathcal{D}, \mathcal{J} \rangle$ in $\mathcal{L}^*$ is defined as *universally valid* iff for every S and every v, $\langle \mathcal{D}^v, \mathcal{J}^v \rangle$ is S-valid. Again we can prove:

[27] See Prawitz 1973, p .236; 1974, p. 73, and this volume. Prawitz does not consider extensions of atomic systems S.

Proposition *Let $\langle \mathcal{D}, \mathcal{J} \rangle$ be in $\mathcal{L}^*$. Then $\langle \mathcal{D}, \mathcal{J} \rangle$ is universally valid iff $\langle \mathcal{D}, \mathcal{J} \rangle$ is S_0-valid.*

This means that we can continue to use the term "valid" (now with respect to some $\mathcal{J}$) interchangeably for both universal and S_0-validity.

It is obvious how to define notions of strict and strong S-validity for $\mathcal{L}^*(S)$ and of strict and strong validity for $\mathcal{L}^*$. We can also prove normalization and strong normalization from strict and strong validity, respectively. I cannot enter these issues here in detail.

The basic difference between derivations and arguments is, of course, that soundness no longer holds in every case; it simply depends on the justifications provided, as was intended by introducing the general notion of argument.

Returning to our previous example, we can now make precise what is meant by the validity of the one-step derivation

$$(\star)\ \frac{A \to (B \to C)}{B \to (A \to C)}\ .$$

This derivation is in fact valid with respect to the standard reductions of implicational logic extended with the reduction given by $(\star\star)$. Now it makes sense to consider derivation structures $\mathcal{D}$ for which *there is* a justification $\mathcal{J}$, such that $\mathcal{D}$ is valid with respect to $\mathcal{J}$. Such derivation structures cannot be represented as derivations in $\mathcal{L}$, as the example of $(\star)$ shows, which is not a rule of $\mathcal{L}$. Therefore *(Compl 1)* does no longer hold. However, we may ask whether *(Compl 2)* holds in the sense that for any derivation structure, which *can be justified* as valid (i.e., which is valid with respect to *some* justification), a derivation in $\mathcal{L}$ of its end formula from its open assumption formulas can be found. That *(Compl 2)* in this form holds is claimed by Prawitz as a conjecture (1973, p. 246), without his being able to indicate how it might be proved.

In addition to *validity* in the sense sketched here, Prawitz also defines a notion of *computability* with respect to arbitrary justifications, which he (unfortunately) calls *strong validity*. It is not surprizing that he is able to establish strong normalization with respect to this general context of arbitrary justifications. We cannot present this here. From our point of view, his general computability concept suffers semantically from the same defect as did the less general computability concept dealt with in sections II.1.1 and II.1.2. Again, Prawitz has to consider irreducible non-canonical arguments as strongly valid, the only difference being that irreducibility is now taken with respect to a justification $\mathcal{J}$, which is not confined to the standard reductions (1973, p. 239; 1974, p. 74).

II.4 The relationship between computability, validity and normalizability: counterexamples

In section II.1.2 we claimed that computability and validity are crucially different, especially by arguing that normal derivations have to be justified semantically. However, at that stage we were not able to give counterexamples establishing this difference, as extensionally the concepts were identical, comprising all derivation in $\mathcal{L}$. Now with respect to the generalized concept of an argument we can provide counterexamples.

We understand the computability of an argument $\langle \mathcal{D}, \mathcal{J} \rangle$, i.e. the computability of $\mathcal{D}$ with respect to a justification $\mathcal{J}$ in the following sense, which leads to weak normalization, and compare it with the validity of $\mathcal{D}$ with respect to $\mathcal{J}$.

Definition of (weak) computability of arguments

(i) A derivation structure of the form $\dfrac{\begin{array}{c}[A]\\ \mathcal{D}\\ B\end{array}}{A{\to}B}$ *is computable with respect to* $\mathcal{J}$*, if for every* $\mathcal{J}' \geq \mathcal{J}$ *and every* $\begin{array}{c}\mathcal{D}'\\ A\end{array}$ *computable with respect to* $\mathcal{J}'$*,* $\begin{array}{c}\mathcal{D}'\\ A\\ \mathcal{D}\\ B\end{array}$ *is computable with respect to* $\mathcal{J}'$*.*

(ii) If a derivation structure $\mathcal{D}$ *is not in I-form and is normal (= irreducible) with respect to* $\mathcal{J}$*, then it is computable with respect to* $\mathcal{J}$*.*

(iii) If a derivation is not in I-form and is not normal with respect to $\mathcal{J}$*, then* $\mathcal{D}$ *is computable with respect to* $\mathcal{J}$*, if* $\mathcal{D}$ *reduces with respect to* $\mathcal{J}$ *to a* $\mathcal{D}'$ *which is computable.*

Counterexample 1: Computability of $\langle \mathcal{D}, \mathcal{J} \rangle$ ***does not imply validity of*** $\langle \mathcal{D}, \mathcal{J} \rangle$

We construct an argument $\langle \mathcal{D}, \mathcal{J} \rangle$ in such a way that $\langle \mathcal{D}, \mathcal{J} \rangle$ is closed, non-canonical and normal, and therefore computable, but not valid.

Choose a closed non-canonical derivation structure $\begin{array}{c}\mathcal{D}\\ A\end{array}$ for non-atomic A, e.g.,

$$\dfrac{\dfrac{\overset{(1)}{[A]}}{A{\to}A}\,(\to I)^{(1)}}{A}\,R \quad \text{with } A \text{ standing for } B{\to}C.$$

Choose $\mathcal{J}$ in such a way that with the rule R no reduction is associated. Then $\langle \mathcal{D}, \mathcal{J} \rangle$ is computable, because it is irreducible. However, $\langle \mathcal{D}, \mathcal{J} \rangle$ is not valid, because it

cannot, as required for validity, be reduced to a canonical derivation structure, since no reduction for $\mathcal{D}$ is available in $\mathcal{J}$.

Counterexample 2: Validity of $\langle\mathcal{D},\mathcal{J}\rangle$ does not imply computability of $\langle\mathcal{D},\mathcal{J}\rangle$

We consider $\langle\bot,\to\rangle$-logic, i.e., a system with a logical constant $\bot$, for which there is no introduction rule. In such a system the derivation $\dfrac{\bot}{A}$, and therefore $\dfrac{\dfrac{[\bot]^{(1)}}{A}}{\bot\to A}\,(1)$ is valid with respect to any $\mathcal{J}$. Now let, for some B, $\mathcal{J}$ be chosen in such a way that $\dfrac{B}{\bot}$ is irreducible. Let $\mathcal{J}$ furthermore be chosen such that $\dfrac{\dfrac{B}{\bot}}{A}$ reduces to itself with respect to $\mathcal{J}$, i.e., the reduction of $\dfrac{\dfrac{B}{\bot}}{A}$ is non-terminating. Then $\dfrac{\dfrac{[\bot]^{(1)}}{A}}{\bot\to A}\,(1)$ is not computable with respect to $\mathcal{J}$, because for computable $\dfrac{B}{\bot}$, $\dfrac{\dfrac{B}{\bot}}{A}$ is not computable (with respect to $\mathcal{J}$).[28]

It can easily be seen that these counterexamples hold for strict validity instead of validity as well, and also for strong validity in comparison with computability when the latter is defined in the strong sense (demanding in clause (iii) that all one-step reductions lead to computable derivations).

It might be added that Counterexample 1 is at the same time a counterexample showing that normalizability, which is implied by validity, does not itself imply validity. Similarly, Counterexample 2 shows that normalizability does not imply computability.

---

[28]The intuitive reason for this behaviour is the following: $\dfrac{\bot}{A}$ is always valid as there is no *closed* valid derivation of $\bot$. However, for open normal derivations $\dfrac{B}{\bot}$ can $\dfrac{\dfrac{B}{\bot}}{A}$ be made non-terminating for appropriate $\mathcal{J}$. (Note that we do not choose $\dfrac{B}{\bot}$ to be just $\bot$, because then the example would not work for strict validity, as $\dfrac{\bot}{A}$ would not terminate.)

The latter is not surprizing as computability is a stronger concept than normalizability, using infinite branching when quantifying over substitution instances of open derivation structures.

II.5 Logical consequence and the validity of inference rules

It is natural that the S-validity of an inference rule

$$\frac{A_1 \quad \ldots \quad A_n}{A}$$

with respect to a justification $\mathcal{J}$ should mean that the one-step derivation structure of the same form is S-valid with respect to $\mathcal{J}$. We can even define the S-validity of an inference rule which allows the discharging of assumptions, such as the generalized rule

$$\frac{\begin{array}{cccc}[C_{11},\ldots,C_{1m_1}] & & [C_{n1},\ldots,C_{nm_n}] \\ A_1 & \ldots & A_n\end{array}}{B}\ .$$

This rule is called *S-valid* with respect to $\mathcal{J}$, if for all $S \geq S_0$, all $\mathcal{J}' \geq \mathcal{J}$, and every list of derivation structures

$$\begin{array}{c}[C_{i1},\ldots,C_{im_i}] \\ \mathcal{D}_i \\ A_i\end{array} \quad (1 \leq i \leq n),$$

which are S-valid with respect to $\mathcal{J}'$, the derivation structure $\dfrac{\mathcal{D}_1 \quad \ldots \quad \mathcal{D}_n}{B}$ is S-valid with respect to $\mathcal{J}'$.

This gives rise to a corresponding notion of consequence. Instead of saying that the rule

$$\frac{A_1 \quad \ldots \quad A_n}{A}$$

is S-valid with respect to $\mathcal{J}$, we may say that A is a *consequence* of $A_1,\ldots,A_n$ *with respect to S and $\mathcal{J}$* ($A_1,\ldots,A_n \models_{S,\mathcal{J}} A$); if we consider universal validity with respect to $\mathcal{J}$, we may speak of *consequence with respect to $\mathcal{J}$* ($A_1,\ldots,A_n \models_{\mathcal{J}} A$); and finally, if there is some $\mathcal{J}$ such that universal validity holds for $\mathcal{J}$, then we may speak of *logical consequence* ($A_1,\ldots,A_n \models A$). Corresponding to the case of rules discharging assumptions, we obtain a notion of consequence

$$\Gamma_1 \Rightarrow A_1,\ldots,\Gamma_n \Rightarrow A_n \models_{S,\mathcal{J}} A$$

for sets of formulas Γ_i. This is to express that the rule

$$\frac{\begin{array}{cc}[\Gamma_1] & [\Gamma_n] \\ A_1 \ \ldots & A_n\end{array}}{B}$$

is S-valid with respect to $\mathcal{J}$, i.e., we have some notion of implication in the antecedent of $\models$, which is independent of whether the logical constant of implication is available in our language. The basic conceptual difference of such a notion of consequence compared to classical consequence is the following. The idea of classical consequence is based on the transmission of truth: A consequence assertion $A_1, \ldots, A_n \models A$ holds if, for any model, if all antecedents A_i are true in the model, then so is the consequent A. In proof-theoretic semantics truth is not just replaced by derivability. It is *not* argued that $A_1, \ldots, A_n \models A$ holds if, for any atomic system S, if all antecedents A_i are derivable with respect to S, then so is the consequent A. Rather, we are saying that $A_1, \ldots, A_n \models A$ holds if, for any atomic system S, every S-valid derivation of $A_1, \ldots, A_n$ *can be extended* to an S-valid derivation of A using the given derivation of $A_1, \ldots, A_n$ *as immediate subderivations* (modulo considering extensions $S' \geq S$ and justifications $\mathcal{J}$). So we are not defining a notion of *constructive truth* alias derivability and then claiming that in consequence constructive truth is being transmitted. Rather, we claim that there is a way, based on formal procedures $\mathcal{J}$, which justifies the derivation of the consequent, if this consequent is appended, as a conclusion of an inference rule, to the derivations of the antecedents. The special feature of proof-theoretic semantics is that consequences arise as final steps of derivations, i.e. that the justification of inference rules and of consequences is exactly the same.

Thus in proof-theoretic semantics, consequence does not mean that the consequent can be derived *from* the antecedents, as might be a naive prejudice. Derivability from assumptions may give rise to a justification which comes to that, but this is not necessarily so. The justification procedure ($\star\star$) of section II.3 for

$$\frac{A{\to}(B{\to}C)}{B{\to}(A{\to}C)}$$

consists essentially of describing a derivation of the conclusion *from* the premiss. But in the case of the more basic procedures like *modus ponens*, this is not the case. Even if it can be shown that every justifiable rule can be derived from the standard rules, this would be just a completeness result like classical completeness — the definition of validity does not refer to derivability according to rules[29].

What goes crucially beyond classical notions of consequence, however, is the notion of implication in the antecedents of consequence statements which can be included

[29] See also Prawitz (1985).

even without implication as a logical constant (i.e., " $\Rightarrow$ " in my terminology). There is some structural notion of implication which is due to the fact that rules can discharge assumptions, comparable to the comma as structural conjunction. This has been used in generalized concepts of inference rules (see section III.1). It is also important for the formulation of a basic sequent calculus in theories of definitional reflection (see section III.3.2).

II.6 Proof-theoretic semantics, proof terms and Martin-Löf's approach

The method of proof terms is a technical device according to which the fact that a formula A has a certain proof can be codified as the fact that a certain term t is of type A, whereby the formula A is identified with the type A. By means of this method, which was introduced by Curry and Howard[30], formulas can be considered as the types of their proofs. This can again be put into a calculus for type assignment, whose statements are of the form $t : A$. A proof of $t : A$ in this system can be read as showing that t codifies a natural deduction proof of A. If t contains variables, $t : A$ may depend on declarations of the form $x_1 : A_1, \ldots, x_n : A_n$, where the $A_1, \ldots, A_n$ correspond to the open assumptions on which the natural deduction derivation of A depends. This idea is exploited in type-theoretical systems such as Martin-Löf's[31] which especially use the idea of *dependent* types, which may contain variables for terms, etc.

Martin-Löf has put this into a philosophical perspective[32] by distinguishing a two-fold sense of proof. First we have proofs of statements of the form

$$t : A \ .$$

These statements are called *judgements*, their proofs are called *demonstrations*. *Within* such judgements the term t represents a *proof* of the *proposition* A. A proof in this sense is also called a *proof object*. So when demonstrating a judgement $t : A$, we demonstrate that a proposition has a certain proof[33]. Within this two-layer system the *demonstration* layer is the layer of argumentation. Unlike proof objects, demonstrations have epistemic significance; their judgements carry assertoric force. The proof layer is the layer at which meanings are explained: The meaning of a proposition A is explained by telling what counts as a proof (object) for A. The distinction made between canonical and non-canonical proofs etc. is a distinction at the propositional and not at the judgemental layer.

---

[30] See de Groote (1995) for references.

[31] See Martin-Löf (1984), Nordström et al. (1990) and Sommaruga (2000).

[32] See Martin-Löf (1995, 1998).

[33] I do not discuss here other forms of judgements which occur in type theory.

On the background of Prawitz's definition of validity, one could expect that Martin-Löf gives a formal definition of validity for proof (objects). However, this is not the case, at least not in the sense of a metalinguistic inductive definition of what is a valid proof (object). Rather, he gives a justification of demonstration steps which refers to the meanings of propositions and to the forms of proof (objects) referred to in its judgements. For example, for the case of implication, the rules for judgements of the form $t : A$ are the following (the standard typing rules of the typed *lambda*-calculus):

$$\frac{\begin{array}{c}[x : A]\\ t(x) : B\end{array}}{\lambda x.t(x) : A{\to}B}\ (\to I) \qquad \frac{t : A{\to}B \quad t' : A}{tt' : B}\ (\to E)$$

where $t(x)$ denotes a term in which x may occur free, and tt' denotes the term application of t to t'.

According to Martin-Löf the justification runs roughly as follows:
$(\to I)$ The proof (object) $\lambda x.t(x)$ is in canonical form, so it is a proof of $A{\to}B$, provided for every proof (object) t' of A, $t(t')$ is a proof of B. The latter holds, as the demonstration of $t(x) : B$ from $x : A$ convinces us exactly of this fact, namely that $t(t')$ proves B if t' proves A.
$(\to E)$ Suppose demonstrations of $t : A{\to}B$ and $t' : A$ are given. Then they convince us that t is a proof (object) of $A{\to}B$ and t' one of A. As a proof of $A{\to}B$, t is already in canonical form or reduces to a proof in canonical form. In each of the two cases, tt' reduces to a proof in canonical form, i.e., tt' is a proof of B.[34]

By justifying (making evident) demonstration steps, Martin-Löf establishes the means for validating proofs. If I have demonstrated $a : A$, I have shown that a is in fact a proof of A. So in a sense Martin-Löf also defines what it means for a proof to be valid (i.e., to be a "real" proof). But this definition is given in the form of the explanation and justification of a system for the demonstration of validity. The crucial difference to Prawitz's procedure is that it is not *metalinguistic* in character, where metalinguistic means that candidates of proofs are specified first and then, by means of a definition in the metalanguage, it is fixed which of them are valid and which are not. Rather, proofs come into play only in the context of demonstrations. I give a proof of A by presenting an object a, of which I demonstrate that it is a proof of A.

Presenting and validating a proof takes place at the same level. Not the proof itself has epistemic force, but its validation in form of a demonstration endows it with epistemic force. Conversely this means that making an assertion, i.e. using epistemic

[34]To make this fully precise, we would have to take equality judgements into consideration, which govern the reductions of proof objects. In a sense, the application operation (indicated here by concatenation) is a kind of justifying operation which, when applied to a canonical proof, yields a proof, as the equality rules for the evaluation of applications (which correspond to β-reduction) show.

force, includes presentation of the proof as a proof object. So when asserting something, I am not just relying on a proof in the sense that I *can* justify it, if I am asked to do so, but I am presenting it as something which by my very reasoning turns out to be a proof (and is as such justified). Proving something and demonstrating the validity of this proof cannot be separated.

This implies a certain expliciteness requirement. When I prove something, I not only have to have a justification for my proof *at my disposal* as in Prawitz, but at the same time have to be *certain* that this justification fulfils its purpose, which is much more. This certainty is guaranteed by a demonstration.

It might be said that Martin-Löf's theory combines the "semantics of proofs" with the "semantical proofs" approaches. In the sense that proofs are treated as objects, he is definitely an adherent of the first one. On the other hand, by rejecting a metalinguistic approach to validity and instead proposing an approach based on the "intuitive" justification of rules, he also represents the second, where "proof" has to be replaced with "demonstration". In Martin-Löf we find both a "semantics of proofs" and "semantical demonstrations".

Part III. Semantical rules and definitional reflection

III.1 Proof-theoretic semantics based on generalized rules

The approach to proof-theoretic semantics described was based on the idea that the candidates for validity are proofs in the sense of derivations, derivation structures or arguments. A different approach would not start with proofs as a whole (which, in a sense, means proofs as objects), but with inference rules and their justification. A proof would then be valid if it proceeds according to justified rules. The idea behind such an approach is that certain inference steps are *semantical* steps, and a valid proof is one which proceeds by such semantical steps. One might characterize the difference between the two approaches by distinguishing between a *semantics of proofs* and *semantical proofs*.

This goes along with a different perspective: Whereas the first approach looks at proofs globally, insisting on features like reducibility to certain forms, the second one considers local steps. The advantage of the second approach is that it can distinguish between local and global features by keeping the individual inference steps distinct from the way these steps are composed to a global proof. Apart from the basic conceptual difference, this makes it very flexible and capable to deal with more sophisticated phenomena than the global approach, such as, e.g., circular reasoning.

One idea is to justify inference rules as following a general semantical pattern. This was attempted by Schroeder-Heister (1981, 1984a).[35] Within a programme of developing a general schema for rules for arbitrary logical constants it was proposed that a proposition of the form $\alpha(A_1, \ldots, A_n)$, where α is an n-ary logical connective, should express what was called the *common content* of *systems of rules*. Here a system of rules is a list of expressions $R_1, \ldots, R_m$, where each R_i is called a "rule". A *rule* R is either a formula A or has the form $R_1, \ldots, R_n \Rightarrow A$, where $R_1, \ldots, R_n$ are themselves rules. In a sense, systems of rules are expressions of a conjunction-implication-calculus (with the comma expressing conjunction and the rule arrow $\Rightarrow$ expressing implication), where implication is iterated only to the left. Instead of systems of rules, of which a single rule is a limiting case, I shall also speak of *conditions*[36]. They are denoted by capital Greek letters Γ, Δ, with and without indices. The derivability of formulas from conditions is explained in the following way: A can be derived from itself considered as a condition (consisting just of A as a premiss-free rule). If $A_1, \ldots, A_n$ have been derived from $\Gamma_1, \ldots, \Gamma_n$, respectively, then B can be derived from $\Gamma_1, \ldots, \Gamma_n$ together with the rule $A_1, \ldots, A_n \Rightarrow B$. This motivates the "rule"-terminology: B can be derived using the

[35] Stimulated by earlier ideas of Prawitz (1978). See also Schroeder-Heister (1987).

[36] This is the terminology proposed by Hallnäs in the context of definitional reflection (see below section III.3.2).

rule $A_1, \ldots, A_n \Rightarrow B$ (and perhaps further conditions) as an assumption. This gives rise to even more complicated rules, permitting to discharge assumptions. For example, the rule $((A \Rightarrow B) \Rightarrow C) \Rightarrow D$ allows one to pass over from C to D, provided C has been derived using $A \Rightarrow B$ as an assumption. This again means that the assumption $A \Rightarrow B$ may be discharged when passing from C to D. This explanation can be extended to rules with individual variables and quantifiers (see Schroeder-Heister 1984b).[37]

Now the semantical idea of the common content of conditions is defined as follows.

Definition (Common content)
The formula A expresses the common content of conditions $\Delta_1, \ldots, \Delta_n$, iff for every condition Γ and every formula C, it holds that

$\Gamma, A \vdash C$ *iff* *for every* i $(1 \leq i \leq n)$, $\Gamma, \Delta_i \vdash C$,

i.e., iff from A, together with possible assumptions Γ, everything follows which follows from each of the conditions Δ_i.

Obviously, *common content* means the same as *content*, if there is just one condition Δ available, where "content" is understood as "set of consequences".

For the standard logical constants this means the following, where conditions, which are lists of rules, are included in braces:

$A \wedge B$ *expresses the common content of* $\langle A, B \rangle$
$A \rightarrow B$ *expresses the common content of* $\langle A \Rightarrow B \rangle$
$A \vee B$ *expresses the common content of* $\langle A \rangle$ *and* $\langle B \rangle$
$\bot$ *expresses the common content of the empty list of conditions (case* $n = 0$*)*

It can then easily be shown that the standard introduction and elimination inferences just express this semantical condition, i.e., they hold iff this condition is fulfilled.

In general, we assume conditions $\Delta_1(p_1, \ldots, p_n), \ldots, \Delta_m(p_1, \ldots, p_n)$ to be associated with an n-ary logical constant α, where every $\Delta_i(p_1, \ldots, p_n)$ contains no propositional variables beyond the indicated $p_1, \ldots, p_n$. Then we suppose that for all $A_1, \ldots, A_n$,

$\alpha(A_1, \ldots, A_n)$ *expresses the common content of the conditions*
$\Delta_1(A_1, \ldots, A_n), \ldots, \Delta_m(A_1, \ldots, A_n)$.

The corresponding introduction inferences in the general case are

$$\frac{\Delta_1(A_1, \ldots, A_n)}{\alpha(A_1, \ldots, A_n)} \quad \ldots \quad \frac{\Delta_n(A_1, \ldots, A_n)}{\alpha(A_1, \ldots, A_n)}$$

[37] For various systems for the handling of rules see Schroeder-Heister (1987).

whereas the elimination rule follows the pattern

$$\frac{\alpha(A_1,\ldots,A_n) \qquad \begin{matrix}[\Delta_1(A_1,\ldots,A_n)] \\ C\end{matrix} \quad \ldots \quad \begin{matrix}[\Delta_n(A_1,\ldots,A_n)] \\ C\end{matrix}}{C}\,.$$

As a corrollary we obtain operator completeness in the sense that any $\alpha(A_1,\ldots,A_n)$ which expresses the common content of certain conditions, can be explicitly defined using the standard connectives $\wedge$, $\vee$, $\rightarrow$ and $\bot$. For example, if $\alpha_1(A_1,\ldots,A_4)$ expresses the common content of $\langle A_1 \Rightarrow A_2\rangle$ and $\langle A_3 \Rightarrow A_4\rangle$, which means that it can be characterized by the following introduction and elimination rules

$$\frac{\begin{matrix}[A_1] \\ A_2\end{matrix}}{\alpha_1(A_1,A_2,A_3,A_4)} \quad \frac{\begin{matrix}[A_3] \\ A_4\end{matrix}}{\alpha_1(A_1,A_2,A_3,A_4)} \quad \frac{\alpha_1(A_1,A_2,A_3,A_4) \qquad \begin{matrix}[A_1 \Rightarrow A_2] \\ C\end{matrix} \qquad \begin{matrix}[A_3 \Rightarrow A_4] \\ C\end{matrix}}{C}\,,$$

then $\alpha_1(A_1,A_2,A_3,A_4)$ can be explicitly defined as $(A_1{\rightarrow}A_2){\vee}(A_3{\rightarrow}A_4)$.

It is, of course, easily possible to define computability and validity notions for derivations based on operators with generalized introduction and elimination rules. However, this would be missing the point, which is the semantical justification of both introduction and elimination rules via the notion of common content of conditions, rather than just a generalized schema for introduction and elimination rules for which then validity could be defined.

Originally, this approach was intended as a semantical approach to logical operators which is well-founded in the following sense. We start with conditions containing no operators at all, calling them *elementary conditions*. In the next step we use conditions referring to logical constants defined by elementary conditions, and so on. For example, we might define a ternary logical constant α_2 with reference to the condition $\langle p_1 \Rightarrow p_2{\vee}p_3\rangle$ with the introduction and elimination rules

$$\frac{\begin{matrix}[A_1] \\ A_2{\vee}A_3\end{matrix}}{\alpha_2(A_1,A_2,A_3)} \quad \frac{\alpha_2(A_1,A_2,A_3) \qquad \begin{matrix}[A_1 \Rightarrow A_2{\vee}A_3] \\ C\end{matrix}}{C}\,.$$

Obviously, $\alpha_2(A_1,A_2,A_3)$ is explicitly definable as $A_1{\rightarrow}(A_2{\vee}A_3)$. Actually, α_2 is an example of an operator which *cannot* be expressed as the common content of elementary conditions. Another example would be the negation operator $\neg$ characterized by the (non-elementary) condition $\langle p \Rightarrow \bot\rangle$.

However, this approach can easily be extended by relaxing the restrictions for conditions, even allowing for circular characterizations. If we want a nullary logical operator

α_3 to express the content of itself, i.e., of $\langle\alpha_3\rangle$, we would obtain introduction and elimination rules like

$$\frac{\alpha_3}{\alpha_3} \qquad \frac{\alpha_3 \quad \overset{[\alpha_3]}{C}}{C} \ ,$$

where the elimination rule is equivalent to

$$\frac{\alpha_3}{\alpha_3} \ ,$$

i.e., we could as well do without any rule for α_3. Although the characterization of α_3 is circular, it fits into our general schema of meaning giving rules based on the notion of common content. It is not suitable for an inductive definition of the validity of arguments, but this is just the point here. We are no longer defining a global concept like validity for derivation structures, but just local rules based on certain requirements. One has then only to be careful how such "non-standard" rules are composed to yield derivations or arguments.

For example, consider defining α_4 be means of its own negation, i.e., by reference to the content of $\langle\alpha_4 \Rightarrow \bot\rangle$. Then the introduction an elimination rules would be

$$\frac{\overset{[\alpha_4]}{\bot}}{\alpha_4} \qquad \frac{\alpha_4 \quad \overset{[\alpha_4 \Rightarrow \bot]}{C}}{C} \ ,$$

where the elimination rule is equivalent to

$$\frac{\alpha_4}{\bot} \ .$$

Allowing such a construction would make the system inconsistent as the following derivation shows:

$$\dfrac{\dfrac{\dfrac{\overset{(1)}{[\alpha_4]}}{\bot}\,(1)\ \alpha_4\text{-I}}{\alpha_4}\,\alpha_4\text{-E}}{\bot}\,\alpha_4\text{-E}$$

However, this inconsistency is achieved by means of a derivation which is not normal in the sense that an application of an introduction rule is followed by that of an elimination, and there is no way to reduce it to a normal derivation. This can be seen as follows. The main reduction step for α_4 would be that

$$\dfrac{\dfrac{\begin{array}{c}(1)\\[\alpha_4]\\ \mathcal{D}\\ \bot\end{array}}{\alpha_4}\,(1)}{\bot} \quad \text{reduces to} \quad \dfrac{\begin{array}{c}(1)\\[\alpha_4]\\ \mathcal{D}\\ \bot\end{array}}{\alpha_4}\,(1)\;\begin{array}{c}\mathcal{D}\\ \bot\end{array}$$

which in the present case, where $\mathcal{D}$ is just $\dfrac{\alpha_4}{\bot}$, means that the original derivation reduces to itself[38], i.e., reduction does not terminate.

These analyses of α_3 and α_4 exemplify investigations of circular reasoning based on the analysis of local rules, which cannot be carried out so easily in a global framework of the validity of derivation structures. I shall resume the subject of circular reasoning in the last section of this paper.

What can clearly be seen is that the approach based on the common content of conditions relies on some sort of basic or primitive logic. If a proposition should express the common content of a system of conditions, there must be ways of expressing conditions and inference rules dealing with that. In the original development of this approach this was considered a calculus dealing with rules of higher levels, which can be read as a system for left-iterated conjunction-implication logic (see Schroeder-Heister 1987). Realizing that there is some basic logic, we might consider a system of *full* conjunction-implicaton logic. This would make clear that there is some sort of logic that comes before semantical justification, which, in terms of first-order logic, is positive logic without disjunction and existential quantification. So what needs semantical justification is basically disjunction and existential quantification, which corresponds to the fact that the notion of common content mimicks the pattern of indirect elimination rules for these logical constants.

Spelling out this basic logic would make it possible to make its substrutural features explicit. The result would be a logical framework based on substructural logic including

---

[38]The reduction step is obtained from the fact that $\dfrac{\alpha_4}{\bot}$ results from $\dfrac{\alpha_4 \quad \alpha_4}{\bot}$, with the same derivation leading to the major and minor premisses of this inference. The reduction for

$$\dfrac{\dfrac{\begin{array}{c}(1)\\[\alpha_4]\\ \mathcal{D}\\ \bot\end{array}}{\alpha_4}\,(1) \quad \begin{array}{c}\mathcal{D}'\\ \alpha_4\end{array}}{\bot} \quad \text{would yield} \quad \begin{array}{c}\mathcal{D}'\\ \alpha_4\\ \mathcal{D}\\ \bot\end{array}\text{, which in the case that } \mathcal{D}\text{' is just } \dfrac{\mathcal{D}}{\alpha_4}\text{, is the same as } \dfrac{\dfrac{\begin{array}{c}(1)\\[\alpha_4]\\ \mathcal{D}\\ \bot\end{array}}{\alpha_4}\,(1)}{\begin{array}{c}\mathcal{D}\\ \bot\end{array}}\text{.}$$

a basic notion of implication which belongs to this framework[39].

It might be noted that the idea that there is some sort of logic prior to semantical justification is not a particular feature of the rule-based approach. Even with the global proof-based approach one would have to spell out the possible form premisses of introduction inferences can take in order to function in canonical proofs, particularly if one goes beyond the standard connectives. This is normally hidden by the fact that just standard connectives and no substructural issues are taken into consideration.

To explain just disjunction and existential quantification may be a disappointingly little task for proof-theoretic semantics. However, this impression is wrong as not only logical constants may be considered within proof-theoretic semantics. It was logic programming which made this idea plausible.

III.2 The challenge from logic programming

The discussion within proof-theoretic semantics has nearly always focused on logical constants. The only exceptions are Lorenzen's operative logics, where the justification of logical rules is embedded in a theory dealing with arbitrary rules, and Martin-Löf's theory of iterated inductive definitions where introduction and elimination rules for atomic formulas are proposed.

The rise of logic programming changed this perspective. In logic programming we are dealing with program clauses of the form

$$A \Leftarrow B_1, \ldots, B_n$$

which *define* atomic formulas. Such clauses can naturally be interpreted as describing introduction rules for atoms. This is quite natural for a PROLOG programmer who reads clauses as rules, although this reading is blurred by the understanding of clauses as disjunctions of the form $\neg A \vee B_1 \vee \ldots \vee B_n$, which is common in treatments of logic programming within the framework of classical logic.

However, if one takes the "rule"-interpretation of clauses seriously, one is inevitably led to a proof-theoretic treatment, which reads programs as collections of introduction rules. Such an approach has been carried out in detail in Hallnäs & Schroeder-Heister (1990, 1991)[40]. It has especially led to extensions of definite Horn clause programming by considering iterations of the rule arrow in bodies of clauses, and, correspondingly, to a natural treatment of negation.

From the point of view of proof-theoretic semantics the following two issues are crucial:

---

[39]Cf. Schroeder-Heister (1991b). The idea of using some sort of substructural implication goes beyond Sambin et al.'s (2000) *Basic Logic*.

[40]See also Schroeder-Heister (1991a).

(1) Introduction rules are available for atoms as well as for logically compound formulas. Logically compound formulas are not distinguished from atoms with respect to their semantical characteristics. They are only distinguished at a higher level as part of a basic logic of conditions, which concerns the way bodies of clauses are built up. In standard logic programming this contains just conjunction expressed by the comma. In extended versions of logic programming it might be more powerful, including implication and even disjunction (in disjunctive logic programming).
(2) The rules one is dealing with are not necessarily well-founded. It is not even required that all possible instances of an atom are defined. For example, the clauses

$$\begin{aligned} p(a,x) &\Leftarrow q(a), r(x), p(a,f(x)) \\ p(x,b) &\Leftarrow s(b), r(x) \end{aligned}$$

are appropriate as program clauses, although there is only a partial definition of p (namely for instances of the forms $p(a,\cdot)$ and $p(\cdot,b)$), and although they include a non-terminating recursion. According to the approach followed in logic programming there is absolutely no point in asking whether syntactically correct program clauses are well-formed with respect to semantic considerations. So logic programming proclaims a great deal of *definitional freedom* in generating programs.

From these observations we learn the following:
(Ad 1) Interpreting logic programming proof-theoretically motivates an extension of proof-theoretic semantics to arbitrary atoms, which yields a semantics with a much wider realm of applications. Conversely, such a proof-theoretic semantics leads to interesting extensions of logic programming, as such a semantics generates elimination rules corresponding to introduction rules given by clauses, which can be successfully exploited in logic programming. This programme was carried out in the extended logic programming language GCLA (Aronsson et al. 1990).
(Ad 2) Using arbitrary clauses without further requirements leads to the idea to follow the same approach in proof-theoretic semantics, admitting just any sort of introduction rules and not just those of a particular form. This idea, which takes definitional freedom over to semantics, is spelled out in the theory of definitional reflection.

Some final remarks concerning Lorenzen's (1955) and Martin-Löf's (1971) ideas seem appropriate, as they both deal with introduction rules for atoms.
Ad Lorenzen Lorenzen's approach is very near to what is done in logic programming as far as the declarative aspects are concerned, in particular his starting from production rules for atoms. His inversion principle is closely related to the more general principle of definitional reflection dealt with in the next section.
Ad Martin-Löf Although Martin-Löf shares with ideas in logic programming and definitional reflection the fundamental idea that atomic formulas can be treated similarly to logical constants, his elimination rules for atoms fundamentally differ from those

considered in the following. This is particularly clear from his treatment of mathematical induction which for him is described by the elimination rule corresponding to introduction rules for a natural number predicate.

The idea of considering introduction rules as meaning-giving rules for atoms is closely related to the theory of inductive definitions in its general form, where inductive definitions are nothing but systems of production rules. And as a definition is the classic way of endowing something with a meaning, it is very natural to include such rules in proof-theoretic semantics.

III.3 Definitional reflection

Proof-theoretic semantics based on definitional reflection is the approach defended by the author of this paper. Like the one based on the common content of conditions it is primarily concerned with the justification of rules or argument steps rather than the validity of derivations or arguments. However, whereas the common-content approach, in its original setting, still put much emphasis on proving global features of derivations, definitional reflection stresses the local character of rules in a very strong sense. Of the ingredients of definitional reflection, it is fair to say that the idea of generalized elimination rules was developed by Schroeder-Heister (though restricted to logical constants), whereas the idea of the strict local reading of rules together with a novel concept of assumptions in logic goes back to Hallnäs. The idea of using logic programming as a model to frame reasoning with atoms was developed independently by both authors. The rest, in particular the technical development, cannot be divided between the two of us. I shall only give a short sketch. There is a presentation of certain basic ideas in this volume by Hallnäs, and there is still a joint monograph project of the two of us.

As was already emphasized, definitional reflection deals with meaning-giving rules for any sort of constants, not just logical ones. It is the non-logical ones which are particularly interesting, as the logical ones have many "nice" features which, in their case, make definitional reflection undistinguishable in many respects from more standard approaches. So it is crucial to stress the *general* character of this sort of proof-theoretic semantics, covering others as special cases.

The local or partial character of rules is expressed by the fact that certain features of rules, which they have as meaning-giving rules, are not *eo ipso* features of the whole derivation in which these rules are embedded. Therefore rules cannot be combined arbitrarily in derivations or, expressed otherwise, combining these rules yields derivations with undesirable properties such as non-normalization.

It turns out that such features can better be expressed using a sequent-style representation of derivations rather than natural deduction format. It can even be claimed

that this is not a matter of convenience but that, in a sense, sequent-style systems are more "natural" than natural deduction.

III.3.1 The philosophical significance of the sequent calculus

In treatments of natural deduction the sequent calculus is often just considered a meta-calculus of natural deduction. According to that reading the sequent $A_1, \ldots, A_n \vdash B$ expresses that a derivation of B from the assumptions $A_1, \ldots, A_n$ is available. However, this does not give justice to the "full" sequent calculus with *left-introduction* rules. In the meta-calculus reading, a rule like

$$\frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\ (\wedge\vdash)$$

should express that a derivation

$$\begin{array}{l}\Gamma \quad A\\ \quad\mathcal{D}\\ \quad C\end{array}$$

can be extended to the derivation

$$\begin{array}{l}\Gamma \quad \dfrac{A \wedge B}{A}\\ \quad\mathcal{D}\\ \quad C\end{array}\quad,$$

and a rule like

$$\frac{\Gamma, A \vdash C \quad \Delta, B \vdash C}{\Gamma, \Delta, A \vee B \vdash C}\ (\vee\vdash)$$

should express that derivations

$$\begin{array}{l}\Gamma \quad A\\ \quad\mathcal{D}_1\\ \quad C\end{array}\quad\text{and}\quad\begin{array}{l}\Delta \quad B\\ \quad\mathcal{D}_2\\ \quad C\end{array}$$

can be extended to a derivation

$$(\star)\quad \frac{\begin{array}{lll}A \vee B & \begin{array}{l}\Gamma \quad [A]\\ \quad\mathcal{D}_1\\ \quad C\end{array} & \begin{array}{l}\Delta \quad [B]\\ \quad\mathcal{D}_2\\ \quad C\end{array}\end{array}}{C}\ .$$

However, although the second derivation is always a natural deduction derivation if the first is one, this is not an operation on natural deduction derivations reflecting the order of deriving propositions.

If we want to turn left-introduction rules of the sequent calculus into a genuine conceptual idea, we have to make special sense of this operation. The idea is to read,

e.g., $(\vee\vdash)$ as *introducing* the assumption $A\vee B$. So $(\vee\vdash)$ is understood as follows: If I have derived C from both Γ, A and Δ, B, then, *by introducing the assumption* $A\vee B$, I can derive C from Γ, Δ, and similarly for other connectives. Thus left-introduction rules of the sequent calculus are understood as assumption-introduction rules.

This is a novel concept of deduction. According to this concept, we not just start from arbitrary assumptions and then assert something by means of specific rules, but it is possible that, in the course of a derivation, we assume something in a specific way. There is not just the trivial assumption rule expressed by initial sequents of the form $A \vdash A$ or $\Gamma, A \vdash A$, but also assumption rules for each connective. The relation between assuming and asserting is completely symmetric. In both cases we have one unspecific rule (initial sequents, by which we both assume and assert a formula) and many specific rules (right-introduction rules for asserting, left-introduction rules for assuming a formula governed by a certain connective).

This idea is highly original and makes the sequent calculus a conceptual device of its own rather then something for whose understanding one has to rely on natural deduction formulations. I do not think that this has been philosophically appreciated in an appropriate way. Of course it is possible to give it a natural deduction rendering. But this is not self-explaining at all. Compare again the $(\vee\vdash)$ rule of the sequent calculus and the $\vee$-elimination rule of natural deduction:

$$(\vee\vdash)\ \frac{\Gamma, A \vdash C \quad \Delta, B \vdash C}{\Gamma, \Delta, A\vee B \vdash C} \qquad (\vee E)\ \frac{\begin{array}{ccc} & [A] & [B] \\ A\vee B & C & C \end{array}}{C}$$

The usual reading of $(\vee E)$ is that from derivations of the premisses we obtain one of the conclusions, i.e.,

$$(\star\star)\ \frac{\begin{array}{ccc} & [A] & [B] \\ \mathcal{D} & \mathcal{D}_1 & \mathcal{D}_2 \\ A\vee B & C & C \end{array}}{C}\ .$$

What is intended, however, is $(\star)$, i.e., given derivations $\begin{array}{c} A \\ \mathcal{D}_1 \\ C \end{array}$ and $\begin{array}{c} B \\ \mathcal{D}_2 \\ C \end{array}$, we obtain a derivation of C *by assuming* $A\vee B$. The difference between $(\star)$ and $(\star\star)$ is that in $(\star)$, the $\mathcal{D}$ is lacking because, in $(\star)$, $A\vee B$ is just a "free-standing" assumption, not the end formula of a subderivation. Of course we can read $(\vee E)$ as expressing $(\star)$ rather than $(\star\star)$, but this inevitably leads to a sort of natural deduction different from what one is used to. If one spells this out one obtains what might be called a "natural-deduction-style sequent calculus", quite dual to the common "sequent-style natural

deduction calculus" (where one uses introduction and elimination rules which operate on the right side of sequents).[41] This shows that the full sequent calculus carries a basic conceptual idea, even if Gentzen himself might only have seen its technical advantage.

Now also the cut rule

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}$$

receives a new conceptual content: It says that the *assumption* A according to the right premiss may at the same time be conceived of as an inferred consequence according to the left premiss, and that these roles coincide. Formulated in natural-deduction-style sequent calculus, it says that from $\begin{array}{c}\Gamma\\ \mathcal{D}\\ A\end{array}$ and $\begin{array}{c}A\ \ \Delta\\ \mathcal{D}'\\ B\end{array}$, we may infer $\begin{array}{c}\Gamma\\ \mathcal{D}\\ A\ \ \Delta\\ \mathcal{D}'\\ B\end{array}$, which is not trivial, as the assumption A might have been inferred by an assumption-introduction rule like $(\star)$. So what is trivial in standard natural deduction may loose its being trivial when considering natural-deduction-style sequent calculus, and has to be stated as an explicit principle. One might call this sort of transitivity principle "natural-deduction-style cut".

This makes the sequent calculus very suitable for the present investigations. Since the information given by certain inference rules is just local, i.e., concerns a single step or a couple of steps, it is not self-evident and not always the case that some "nice" global behaviour expressed by cut comes out of composing them. Putting this the other way round we may say: Framing things in terms of the sequent calculus makes it possible *to keep certain information local* in a natural way. We can justify inference rules for sequents which can be combined in the sense of the sequent calculus (without cut), but which still leave open whether corresponding natural-deduction derivations can be combined, which depends on the availability of cut elimination.

In this sense the sequent calculus gives a very fine-grained analysis of deduction, keeping apart derivation pieces and the joining of such pieces. This opens new perspectives of logical modelling, as we can now distinguish between cut holding globally, cut holding only locally, cut holding for certain propositions etc. The fact that for standard systems of logic cut is eliminable is more than a mere technical result from which

[41] This is to be distinguished from what Negri & v. Plato (2001a) call a "sequent calculus in natural deduction style", which is a calculus of sequents which imitates certain features of natural deduction concerning vacuous and multiple discharging of assumptions. However, some proof-theoretic investigation of what we call "natural-deduction-style sequent calculus" (which formally is a natural deduction system) can be found in v. Plato (2001). There one also finds, as a natural deduction rule, the left-introduction rule for $\rightarrow$ of the sequent calculus, which we cannot discuss here.

consistency etc. can be concluded, but shows that combining local information yields global information. We are now in a position to consider less well-behaved systems in which we do not have this feature in full generality.

III.3.2 Rules of definitional reflection

As with the common-content conception presented in section III.1, we need a basic logic of *conditions* in terms of which the meanings of certain expressions are explained. In the propositional case, this will be a conjunction-implication logic in a sequent-style setting. In the common-content case we used a special notation for conditions as they were interpreted as systems of rules, with the rule arrow $\Rightarrow$ expressing an implication which may be iterated to the left and the comma standing for the association of rules and corresponding to conjunction. Here we shall use a full conjunction-implication-calculus in the standard notation using $\rightarrow$ and $\wedge$ as connectives, which may be arbitrarily nested (i.e., with implication not just iterated to the left). So one has to bear in mind that conjunction-implication-formulas denote expressions of the basic logic of conditions which are to be distinguished from the expressions whose meaning is to be explained by specific rules. With respect to the basic logic, these expressions are all atomic, although they may be internally composed in various ways, i.e., contain predicates or functions and arguments. In particular, they may be logically composed, where then logical operators have to be notated in a different way, e.g. by using $\supset$ for implication etc. These logical operators are then functions occurring within atoms. This corresponds to the way taken in logic programming where all expressions to be defined in a program are considered as atoms, whereas the basic logic describes the way clauses are written and handled.

Given a set $\mathcal{A}$ of atoms, *conditions* over $\mathcal{A}$ (or just conditions (*simpliciter*), if $\mathcal{A}$ is clear from the context) are defined as formulas built up from $\mathcal{A}$ by means of conjunction $\wedge$ and implication $\rightarrow$. The rules of the *basic logic of conditions* $\mathcal{C}_{\mathcal{A}}$ over $\mathcal{A}$ are then the following ones.

$$\Gamma, A \vdash A$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C} \quad \frac{\Gamma, B \vdash C}{\Gamma, A \wedge B \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \qquad \frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C}$$

This system becomes a system of proof-theoretic semantics by adding rules for handling definitions. Just as in logic programming, a definition is a set of clauses of the form

$$a \Leftarrow C$$

where a is an atom and C a condition. For the time being we just consider the case without individual variables occurring in a or C.

If $\mathbb{D}$ is a definition, the right-introduction rules for defined atoms are

$$(\vdash a)\ \frac{\Gamma \vdash C}{\Gamma \vdash a} \qquad \text{if } a \Leftarrow C \text{ occurs in } \mathbb{D}\ .$$

The left-introduction rules run as follows: Suppose

$$\begin{cases} a & \Leftarrow & B_1 \\ & \vdots & \\ a & \Leftarrow & B_n \end{cases}$$

is the set of clauses defining a in $\mathbb{D}$. Then the left-introduction rule for a is

$$(a\vdash)\ \frac{\Gamma, B_1 \vdash C \quad \ldots \quad \Gamma, B_n \vdash C}{\Gamma, a \vdash C}$$

The intuitive meaning of this rule is the following: Everything that follows from every possible definiens of a, follows from a itself. It is called the *principle of definitional reflection*, as it reflects upon the definition as a whole. If $B_1, \ldots, B_n$ exhaust *all possible conditions* to generate a according to the given definition, and if each of these conditions entails the very same conclusion, then a itself entails this conclusion. If we talk generically about these rules (i.e., without mentioning a specific a, we shall write $(\vdash\mathbb{D})$ and $(\mathbb{D}\vdash)$.

The resulting system is called $\mathcal{C}_{\mathcal{A}}(\mathbb{D})$ or $\mathcal{C}(\mathbb{D})$, as $\mathcal{A}$ is clear from the context. If $\Gamma \vdash A$ is derivable in $\mathcal{C}(\mathbb{D})$, we also write $\Gamma \vdash_{\mathbb{D}} A$. It is obvious that $\mathcal{C}(\mathbb{D})$ is non-monotonic in $\mathbb{D}$ in the sense that if $\mathbb{D}$ is extended with an additional clause

$$a \Leftarrow B_{n+1}$$

for a, then previous applications of the $(\mathbb{D}\vdash)$ rule may fail to remain valid.

The $(\mathbb{D}\vdash)$-rule is closely related to the idea of propositions expressing the common content of conditions presented in the section III.1. In fact, $(\vdash\mathbb{D})$ and $(\mathbb{D}\vdash)$ can be read as capturing this idea for the case of the sequent calculus if one defines the common content of a with respect to $\mathbb{D}$ as the set of pairs $\langle \Gamma, C\rangle$, such that $\Gamma, B_i \vdash C$ can be derived for every condition B_i. However, the crucial difference is that $(\mathbb{D}\vdash)$ is now considered to be a rule *for assuming* a in the sense described above, not as a rule which tells what can be derived given that a *has already been derived*. So the important step is that the idea of the common content is turned into an *assumption rule* (apart from the fact that the restriction to logical constants is given up).

This is technically very simple, but becomes much more complicated if the definition of a is allowed to contain individual variables. In that case unification procedures have to be employed to find out the conditions of a, if a is not defined literally in $\mathbb{D}$, but

a is a substitution instance of something defined in $\mathbb{D}$, or if only certain substitution instances of a are defined in $\mathbb{D}$.[42]

Whether the calculus of definitional reflection admits cut elimination depends on the definition $\mathbb{D}$ one is dealing with. If we have cut elimination with respect to a (i.e., cut elimination for cuts with cut formula a), then $\mathbb{D}$ is called *total* with respect to a, otherwise *partial*.[43] Cut elimination is something which is considered to be established "afterwards", i.e., it is a "factual" feature of the definition, not a prerequisite for something suitable as a definition. In this sense a definition defines something *locally*. Global properties like cut elimination, which depend on the interaction of rules, need not necessarily hold.

This does not mean that the definitional rules are just arbitrary. The rules ($\vdash\mathbb{D}$) and ($\mathbb{D}\vdash$) display a certain *harmony* in the sense that a *single* reduction step for cut elimination can be performed, which may be called a *main reduction step*:

$$\dfrac{\dfrac{\Gamma \vdash B_i}{\Gamma \vdash a} \quad \dfrac{\{\Delta, B_i \vdash C\}_{1\le i\le n}}{\Delta, a \vdash C}}{\Gamma, \Delta \vdash C} \quad \text{reduces to} \quad \dfrac{\Gamma \vdash B_i \quad \Delta, B_i \vdash C}{\Gamma, \Delta \vdash C}$$

i.e., the cut with a is reduced to a cut with the defining condition B_i of a.

Performing such reductions yields a global cut elimination result only if B_i is itself of lower complexity than a and that the corresponding chain of definitions is well-founded, which is, however, not required in every possible case. E.g., for a definition

$$a \Leftarrow a{\to}b$$

a cut with a:

$$\dfrac{\dfrac{\Gamma \vdash a{\to}b}{\Gamma \vdash a} \quad \dfrac{\Delta, a{\to}b \vdash C}{\Delta, a \vdash C}}{\Gamma, \Delta \vdash C}$$

is reduced to a cut with the cut formula $a{\to}b$ of higher complexity:

$$\dfrac{\Gamma \vdash a{\to}b \quad \Delta, a{\to}b \vdash C}{\Gamma, \Delta \vdash C}\ .$$

Whether a chain of cut reductions starting this way terminates, depends entirely on how b is defined in $\mathbb{D}$ (and in particular whether it is defined at all).

---

[42] These problems are discussed at length in Hallnäs & Schroeder-Heister (1991) and Schroeder-Heister (1993, 1994b). They are closely connected with problems Lorenzen had with the original formulation of his inversion principle; see Hermes (1959). Hallnäs (1991) prefers using an infinitary version of conjunction and rules with infinitely many premisses to avoid these problems as long as he is not considering computational issues as they show up in logic programming.

[43] This terminology was chosen by Hallnäs, who borrowed it from recursive function theory to stress the analogy with total and partial recursive functions (see the end of this section).

The harmony of $(\vdash\mathbb{D})$ and $(\mathbb{D}\vdash)$ might therefore be described as permitting *relative cut elimination* in the sense that if a is defined as

$$\left\{\begin{array}{lcl} a & \Leftarrow & B_1 \\ & \vdots & \\ a & \Leftarrow & B_n \end{array}\right. ,$$

then cut holds for a provided it holds for each condition B_i of a, i.e., if cut for a is eliminable in the system to which cut rules for all B_i $(1 \leq i \leq n)$ are added. One might also call this feature *local cut elimination*. It guarantees that there is some conservativeness in the system, i.e., that the definition of a does not create anything new. Again, this is *relative* or *local* conservativeness: The definition of a is conservative provided for each condition of a conservativeness holds. It does not man that the definition $\mathbb{D}$ as a whole is conservative, i.e. that *global* conservativeness holds. We may use cut for a, i.e. take the risk of being non-conservative only if we can take this risk for the conditions of a.

It also holds that we have *relative* or *local* uniqueness in the following sense: If we duplicate the definition $\mathbb{D}$ with A^* being the duplicate of A, then for any a we can derive uniqueness in the sense of $a \dashv\vdash a^*$, *provided* we have uniqueness in the sense of $B_i \dashv\vdash B_i^*$ for every condition B_i of a. This does not necessarily entail global uniqueness for a as the definition of a might not be well-founded.

This is the only, but crucial, restriction on the form of meaning-giving rules: There must be a harmony between $(\vdash\mathbb{D})$ and $(\mathbb{D}\vdash)$ which entails local conservativeness and uniqueness without postulating such features for the system as a whole.

In this way definitional reflection retains basic definitional features at the local level without sacrificing the freedom of formulating definitions, which includes formulating them in a circular way. The analogy between partial and total recursive functions might be taken as a guiding example: A partial recursive function has certain features which make it recursive or computable. However, this does not necessarily mean that for every single argument it yields a definite value. Being total for a partial recursive function is something that may "turn out later"; it is nothing that has to be guaranteed in advance by restricting the possible form of recursive function definitions.

It is an easy task to give a natural deduction version of definitional reflection. The left-introduction rule $(a\vdash)$ for an atom a would then be turned into an a-elimination rule of the form

$$(aE)\ \frac{\begin{array}{cccc} a & [B_1] & & [B_n] \\ & C & \ldots & C \end{array}}{C} .$$

A thorough investigation, which I cannot enter here, would have to discuss the features of a natural-deduction-style sequent calculus in detail.

III.4 Global features of the consequence relation

According to the standard approach to proof-theoretic semantics based on the validity of derivations, a central result would be to prove the completeness of, say, intuitionistic first-order logic with respect to the validity concept. This does not make sense with definitional reflection, since we are no longer considering a global property of derivations ("validity") of which we can ask whether derivations in a certain formal system have this property. Justifying a formal system now means embedding this system into the framework of definitional reflection, i.e., showing that its rules are rules over some definition $\mathbb{D}$. The interesting metalogical results concern the global features of the resulting consequence relation $\vdash_{\mathbb{D}}$ depending on the local features of the definition $\mathbb{D}$. In this section I relate some features of $\mathbb{D}$ with those of $\vdash_{\mathbb{D}}$.

(1) Cut

If $\vdash_{\mathbb{D}}$ is total, i.e., cut holds for $\vdash_{\mathbb{D}}$, then we can use every assumption as something that can be "bought out" by an assertion. No assumption, even when introduced by an assumption rule, i.e. a left-introduction rule, needs to remain an assumption forever. This means that pieces of derivations can be joined together arbitrarily to yield a global derivation without paying attention to how assumptions have been generated, i.e., whether they have been generated without reference to their meaning by using an initial sequent $A \vdash A$, or with reference to their meaning by using a left-introduction rule. If cut does not hold, this means that certain assumptions introduced by left-introduction rules have to be kept as assumptions, even if they can be derived, as the balance between asserting and assuming is only local and relative.

In the following, I mention some cases where, due to $\mathbb{D}$ obeying certain requirements, $\vdash_{\mathbb{D}}$ is in fact total.

$\vdash_{\mathbb{D}}$ is total for definite clauses

Following the terminology in logic programming, a definite clause is a clause which in its body only contains conjunctions of atoms and no implications. This is, for example, the case with logical connectives apart from implication. In propositional logic they can be defined using a truth predicate T as follows:

$$\mathbb{D}\left\{\begin{array}{ll} T(p\&q) & \Leftarrow\ T(p)\wedge T(q) \\ T(p\vee q) & \Leftarrow\ T(p) \\ T(p\vee q) & \Leftarrow\ T(q) \\ \text{no clause for } T(\bot) & \end{array}\right.$$

For $\mathbb{D}$ it can easily be shown that $\vdash_{\mathbb{D}}$ is total. In general, it holds that for definitions $\mathbb{D}$ consisting of definite clauses only (as in standard logic programming), $\vdash_{\mathbb{D}}$ is total (see Hallnäs & Schroeder-Heister 1991).

This does not mean that in all relevant or interesting cases $\vdash_{\mathbb{D}}$ is total, as it is implication which yields particular power. Definite clause programming is a special restriction which does not employ this power. Implications in the bodies of clauses such as in the definition of the logical implication connective

$$T(p \supset q) \quad \Leftarrow \quad T(p) \rightarrow T(q)$$

provide a strong extension of logic programming.

$\vdash_{\mathbb{D}}$ is total for well-founded definitions

A definition $\mathbb{D}$ is well-founded if for any atom a, the chain of definitional predecessors of a ends after finitely many steps. Here an atom b is a definitional predecessor of a if b occurs as a subformula of B, where $a \Leftarrow B$ is an instance of a clause in $\mathbb{D}$. That $\vdash_{\mathbb{D}}$ is total in this case is obvious in view of relative cut elimination. In the definition of the standard logical constants (including implication) we have exactly this situation.

It is the influence of logic programming that motivates us to consider a more general case. Furthermore, it is not in all cases easy to check whether a definition is well-founded or not. Therefore, if one requires well-foundedness as a necessary feature of definitions, one not only excludes onself from powerful definitions but also puts a heavy burden on proving the admissibility of proposed definitions.

$\vdash_{\mathbb{D}}$ is total for contraction-free logics

Suppose the underlying logic of conditions is contraction-free, i.e. the rule of contraction

$$\frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C}$$

is not assumed to hold. Here the antecedent Γ of a sequent $\Gamma \vdash A$ is considered to be a multiset, which means that the order of elements is irrelevant, but the multiplicity of identical elements counts. Then it can be shown that for *any* definition $\mathbb{D}$, the consequence relation $\vdash_{\mathbb{D}}$ is total (see Schroeder-Heister 1992). This result is very interesting as it shows the significance of substructural features in our logic of definitions. A closer inspection shows that the reason why cut may fail in the case with contraction is that contraction allows one to identify propositions introduced as assumptions *according to their meaning*, i.e., by a left-introduction rule, with propositions introduced as assumptions *without reference to their meaning*, i.e., by initial sequents. If these two issues are kept apart, as is the case in a contraction-free system, totality holds.

$\vdash_{\mathbb{D}}$ is total for restricted initial sequents

The remark at the end of the preceding paragraph suggests the following restriction of initial sequents $A \vdash A$ [44]: An initial sequent $A \vdash A$ is only allowed for atomic A, and here only for those A for which there is no left-introduction rule $(a\vdash)$ available. The motivation behind this restriction is the idea that assumptions should only be introduced according to their meaning, i.e., according to the rule of definitional reflection. Now the rule of definitional reflection is always defined, even if there is no clause for a in the definition $\mathbb{D}$. In the latter case definitional reflection yields $a \vdash C$ for any C, i.e., from the undefined anything can be obtained. Absurdity $\bot$ is the characteristic proposition which is undefined. However, we may modify our approach by *singling out* a set of certain atoms $\mathcal{U}$ for which definitional reflection is not allowed, which are simply stipulated as being without defined meaning (i.e., even without the vacuous meaning expressing that there is no right-introduction rule). This corresponds to saying that for any $a \in \mathcal{U}$, the clause $a \Leftarrow a$ is in $\mathbb{D}$ as the only clause defining a, for in that case the left- and right-introduction rules for a are just redundant. Then it can be shown that $\vdash_{\mathbb{D}}$ is total, even with full contraction. This is not surprising as this proposal, rather than restricting contraction, keeps assumptions introduced by initial sequents from the very beginning apart from those introduced by left-introduction rules. It also shows that contractions of an a which is introduced by a left-introduction rule several times at different places in a derivation causes no harm. It establishes that the identification of two *semantically different* ways of introducing a as an assumption causes the failure of totality.[45]

(2) Standard consequence

When defining the validity of derivations, closed derivations were considered as basic, and an open derivation was defined as valid if it yielded a closed valid derivation when its open assumptions were substituted with closed valid derivations. This suggests considering as a standard global feature of a consequence relation $\vdash_{\mathbb{D}}$ that the following holds:

$$(\star)\ \Gamma \vdash_{\mathbb{D}} A \quad \text{iff} \quad (\vdash_{\mathbb{D}} \Gamma \ \text{ implies } \ \vdash_{\mathbb{D}} A)\ .$$

Following Hallnäs (1991)[46] we call the consequence relation $\vdash_{\mathbb{D}}$ *standard*, if $(\star)$ holds.

[44] Everything I am going to say holds for initial sequents $\Gamma, A \vdash A$ as well, which make Thinning an implicit feature.

[45] Kreuger (1994) was the first to consider such a system. The cut elimination result is due to Schroeder-Heister (1994a). This system is related to Jäger & Stärk's (1998) proof-theoretic approach to logic programming and their three-valued semantics.

[46] Slightly modified

In another terminology, which goes back to Lorenzen, we might say: $\vdash_{\mathbb{D}}$ is standard, if for every rule derivability coincides with admissibility.

It turns out that this condition is even stronger than totality. Call $\vdash_{\mathbb{D}}$ *complete*, if for every Γ and A,

$$\Gamma \not\vdash_{\mathbb{D}} A \quad \text{iff} \quad (\vdash_{\mathbb{D}} \Gamma \ \text{ and } \ A \vdash_{\mathbb{D}} \bot)$$

holds, where $\vdash_{\mathbb{D}} \Gamma$ means $\vdash_{\mathbb{D}} B$ for every $B \in \Gamma$, and $\bot$ is absurdity. Call $\vdash_{\mathbb{D}}$ *strongly complete*, if for every Γ and A,

$$\text{either} \ \ \Gamma \vdash_{\mathbb{D}} A \ \ \text{or} \ \ (\vdash_{\mathbb{D}} \Gamma \ \text{ and } \ A \vdash_{\mathbb{D}} \bot)$$

holds. Obviously, in classical logic, "strongly complete" means the same as "complete".

Then the following results can easily be established.

(1) If $\vdash_{\mathbb{D}}$ is total and strongly complete, then $\vdash_{\mathbb{D}}$ is standard.
(2) If $\vdash_{\mathbb{D}}$ is standard, then $\vdash_{\mathbb{D}}$ is total and complete.

From (1) and (2) it follows classically that

$\vdash_{\mathbb{D}}$ is total and strongly complete iff $\vdash_{\mathbb{D}}$ is standard.

(see Hallnäs 1991). This shows that the requirement that consequence be standard, which is built into the notion of validity of derivations, is very strong. It provides another argument for an approach which is more expressive by putting less demands on the consequence relation.[47]

There are many other aspects that we cannot discuss here, as, for example, the precise formulation of the inversion principle (i.e., of $(\mathbb{D}\vdash)$) in the presence of variables, or the question of global properties of consequence restricted to certain propositions a, such as cut with the cut-formula a.

III.5 The circularity example

To demonstrate some particular features of $\vdash_{\mathbb{D}}$ in the case where $\mathbb{D}$ is not standard as in the case of logical constants, we consider, as an instructive example, the case where we have just two atoms a and $\bot$, where $\bot$ is not defined at all, having the *ex falso quodlibet* as its left-introduction rule, and a is defined by the single clause

$$(\mathbb{D}) \quad a \Leftarrow a{\to}\bot \ ,$$

or, using $\neg a$ as an abbreviation for $a{\to}\bot$,

$$a \Leftarrow \neg a \ .^{48}$$

[47] Hallnäs even considers the standard view of implication ("standard" in the above terminological sense) as expressing the impredicative character of implication, as it quantifies over all derivations (see above footnote in section II.1.2, Ad (2)).

Defining something by its own opposite might look strange as it stands. However, it is closely related to a situation which arises in connection with antinomies where one uses similar constructions in order to derive contradictions.

Obviously, this definition is not well-founded. Furthermore, it is not total, as can be seen as follows. Consider the following derivation:

$$(1)\qquad \dfrac{\dfrac{\mathcal{D}\left\{\begin{array}{l} \dfrac{\dfrac{a\vdash a \quad \bot\vdash\bot}{a, a{\to}\bot\vdash\bot}\,(a\vdash)(\star)}{a\vdash\bot}\\[2mm] \hline \vdash a{\to}\bot \end{array}\right.}{\vdash a}\,(\vdash a) \qquad \dfrac{\dfrac{a\vdash a \quad \bot\vdash\bot}{a, a{\to}\bot\vdash\bot}\,(a\vdash)(\star)}{a\vdash\bot}}{\vdash\bot}\,\mathrm{Cut}$$

If we had totality, we could eliminate cut and would be able to derive $\vdash\bot$, which is, however, not possible, as there is no definitional clause for $\bot$. Therefore cut is not admissible and the definition of a is not total.

If we perform a reduction on (1) as in proofs of cut elimination, we have to take into account that at the positions marked with an asterisk, an implicit contraction is taking place. In order to perform a main reduction reducing the given cut to a cut with the premiss $a{\to}\bot$ of a as the cut formula, we have first to perform a cut with the original cut formula a at the place where the contracted formula occurs for the first time, in this case the initial sequent $a\vdash a$, and then a cut with $a{\to}\bot$. This yields the derivation

$$\dfrac{\mathcal{D}\atop\vdash a{\to}\bot \qquad \dfrac{\dfrac{\mathcal{D}\atop\vdash a{\to}\bot}{\vdash a}\quad a\vdash a}{\vdash a}\,\mathrm{Cut} \qquad \bot\vdash\bot}{a{\to}\bot\vdash\bot}\,\mathrm{Cut}\ ,$$

which by a trivial step removing $a\vdash a$ is reduced to

$$\dfrac{\mathcal{D}\atop\vdash a{\to}\bot \qquad \dfrac{\mathcal{D}\atop\vdash a{\to}\bot}{\vdash a}\quad \bot\vdash\bot}{a{\to}\bot\vdash\bot}\,\mathrm{Cut}\ ,$$

[48] Hallnäs (1991) introduced this example as a standard example for nonstandard consequence.

where we have just a cut with the cut formula $a{\to}\bot$ as desired. If we now apply the main reduction for $\to$, we obtain

$$\dfrac{\dfrac{\mathcal{D}}{\vdash a{\to}\bot}\quad\dfrac{a\vdash a\quad\bot\vdash\bot}{a, a{\to}\bot\vdash\bot}}{\dfrac{\vdash a\qquad a\vdash\bot}{\vdash\bot}\,\mathrm{Cut}}\ ,$$

which is exactly the derivation (1) we started with. This means that there are local reductions which turn into a loop when iterated to obtain a global reduction.

This is an example of reasoning where global cut elimination fails. In this sense the consequence relation obtained is only partial. This is exactly what is wanted in this case: We can give a sensible account of circular reasoning without the devastating result of leading to a contradiction. Locally, by considering each single step, everything is correct with circular reasoning. Only globally it lacks certain features, which otherwise would render it inconsistent.

However, there are different analyses possible which do not even allow one to generate derivation (1). If in (1) we check how the cut formula a was introduced as an assumption on the left of the turnstile, it turns out that, at different places, a is introduced by an initial sequent and by the $(a\vdash)$-rule (the rule of definitional reflection for a). These occurrences are then identified by means of (implicit) contraction at the places indicated with an asterisk. Now according to our distinction, made in section III.3, between the unspecific introduction of assumptions by initial sequents and the specific introduction of assumptions by $(\mathbb{D}\vdash)$, we might argue that these different sorts of assumptions must never be identified. In this case, by disallowing contraction, we cannot generate the cut leading to absurdity, because the $(\star)$-steps are not permitted. As mentioned in section III.4, without contraction the system would be total. Another possibility would be to restrict initial sequents in accordance with another point made in section III.4, viz. to prohibit a in an initial sequent because a left-introduction rule for a is available. In that case again cut elimination would hold, and we would not be able to generate the critical cut. I cannot go into further detail here. This just serves to illustrate that by means of definitional reflection we gain a fresh view of circular phenomena and also of antinomies.[49].

If we deal with our example in natural-deduction-style sequent calculus, we would define a derivation to be normal if major premisses of elimination rules occur only as assumptions. A cut would be a major premiss of an elimination rules which is not an

[49]It might be annotated that this sort of analysis is not unrelated to Fitch's and Ackermann's logical analysis of the paradoxes, in particular of Curry's paradox. See Fitch (1936, 1952), Ackermann 1950, Robering (2001)

assumption. This corresponds to the definition of a maximum formula, if only generalized ("indirect") elimination rules are considered, and, after Martin-Löf[50], conclusions of elimination rules, which are at the same time major premisses of elimination rules, are considered maximal.[51] So according to this terminology, being normal means being cut-free. In the present example the derivation of absurdity is not normalizable, i.e., the normalization procedure loops. This is shown as follows, using

$$\frac{a}{\neg a}$$

as the elimination rule for a which is equivalent to

$$\frac{a \quad \begin{matrix}\neg a\\ C\end{matrix}}{C}\ .$$

Then we have the following derivation of $\bot$:

$$\dfrac{\mathcal{D}\left\{\begin{array}{l}(1)\\ \dfrac{[a]}{\neg a}\,(aE)\end{array}\right. \quad \dfrac{\dfrac{\dfrac{}{\neg a}\,(1)}{[a]}\,(\to E) \quad \mathcal{D}}{\dfrac{\bot}{\neg a}\,(\to I)\ (1)} \quad \dfrac{\neg a}{a}\,(aI)}{\bot}\,(\to E)$$

which, when an $\to$-reduction step is applied to the maximum formula $\neg a$ (the left premiss of the last inference), reduces to

$$\dfrac{\dfrac{\dfrac{\mathcal{D}}{\neg a}\,(aI)}{\dfrac{a}{\neg a}\,(aE)} \quad \dfrac{\mathcal{D}}{\dfrac{\neg a}{a}\,(aI)}}{\bot}\,(\to E)$$

Reducing this derivation by removing the maximum-formula a in the left branch gives

$$\dfrac{\mathcal{D} \quad \dfrac{\mathcal{D}}{\dfrac{\neg a}{a}\,(aI)}}{\bot}\,(\to E)$$

which is exactly the derivation we started with. Actually, this derivation is nothing but an expanded version (with explicit negation) of the example concerning α_4 given in section III.1. So as with the sequent calculus presentation, we have an example

[50] according to Prawitz (1971), p. 253seq.

[51] So it is only consequent that Martin-Löf (1971), and, following him, Hallnäs (1991) speak of "cuts" and "main cuts" in natural deduction.

of a derivation of absurdity which loops when reduced.[52] It may also be noted that we have again contraction of different sorts of assumptions playing its part: Consider the two occurrences of assumptions a in $\mathcal{D}$. The left one is a major premiss of the a-elimination rule, whereas the right one is just introduced as an assumption in an unspecific way. In spite of their different functions, they are both discharged at the same time by introducing $\neg a$ and therefore contracted to the same a occurring in the negation $\neg a$.

This example can also be read as a derivation of $\bot$ from $a \leftrightarrow \neg a$ in ordinary intuitionistic propositional logic, if applications of a-introduction and a-elimination are replaced with modus ponens using $\neg a \to a$ and $a \to \neg a$ as assumptions, respectively:

$$\dfrac{\mathcal{D}\left\{\begin{array}{c} \\ \dfrac{a\to\neg a \quad [a]}{\neg a} \\ \\ \end{array}\right. \quad \begin{array}{c} (1) \\ (1) \\ [a] \\ \dfrac{\bot}{\neg a}\,(1) \end{array} \quad \begin{array}{c} \\ \\ \mathcal{D} \\ \dfrac{\neg a\to a \quad \neg a}{a} \end{array}}{\bot}$$

As before, this derivation is not normal, as the left premiss of the last inference ($\neg a$) is a maximum formula. Applying a $\to$-reduction step yields

$$\dfrac{a\to\neg a \quad \dfrac{\begin{array}{c}\mathcal{D} \\ \neg a\to a \quad \neg a\end{array}}{a} \quad \dfrac{\neg a\to a \quad \begin{array}{c}\mathcal{D} \\ \neg a\end{array}}{a}}{\neg a}$$

$$\bot$$

This is a normal derivation in the standard sense. On the other hand, it is quite plausible to consider a piece of derivation of the form

$$\dfrac{B\to A \quad \dfrac{\begin{array}{c}\mathcal{D} \\ A\to B \quad A\end{array}}{B}}{A}$$

not to be normal and rather contract it to $\begin{array}{c}\mathcal{D} \\ A\end{array}$. If this is applied to the previous derivation (with A as $\neg a$ and B as a), then we obtain the original derivation derivation back,

[52] For demonstration purposes, we have tacitly assumed that, as in standard natural deduction, every premiss of an inference step can at the sametime be a conclusion of another inference step. This is not a priori justified if we take the idea of natural-deduction-style sequent calculus seriously, where major premisses of E-rules have to be kept as assumptions. The first reduction step making the premiss of (aE) a conclusion of (aI) could not be carried out under such a restriction. This way of proceeding is analogous to the fact that in the previous sequent calculus example we first assumed cut and then showed which result is obtained when we try to eliminate it by means of reduction.

which means that we have an example of a non-normalizable derivation in intuitionistic propositional logic (with "normalizable" understood in a new, but very plausible sense). This phenomenon was discovered by Ekman (1998). He uses the example to demonstrate (1) that the concept of a normal derivation is not a clearcut one and (2) that intuitionistic implication is not as well-behaved as one usually takes it to be.

A philosophical consequence of this observation is the following. If the logical proof of absurdity $\bot$ from $a \leftrightarrow \neg a$ is flawed in some way, as Ekman's derivation shows, then there is a strong difference between $\bot$ and $a \leftrightarrow \neg a$. The proposition $a \leftrightarrow \neg a$ just means some sort of circularity whereas $\bot$ destroys the whole system as it entails everything. This has further consequences for the treatment of antinomies. Normally antinomies result in a proof of $a \leftrightarrow \neg a$ for some a. If this does not imply absurdity at the same time, antinomical constructions are not as harmful as they are supposed to be, but represent 'just' definitions resulting in circularities. This coincides with the idea of dealing with circular constructions as *phenomena* rather than *problems* to be avoided at any price.[53] It the local approach to proof-theoretic semantics can illuminate such issues, it shows that proof-theoretic semantics is more than just another semantical interpretation of first-order logic.

[53]See, e.g., Barwise & Moss (1996) and references therein.

References

Ackermann, W. (1950). Widerspruchsfreier Aufbau der Logik.I. Typenfreies System ohne *tertium non datur*. *Journal of Symbolic Logic* **15**, 33–57.

Aronsson, M., Eriksson, L.-H., Gäredal, A., Hallnäs, L. & Olin, P. (1990). The programming language GCLA: A definitional approach to logic programming. *New Generation Computing* **4**, 381–404.

Barwise, J. & Moss, L. (1996). *Vicious Circles: On the Mathematics of Non-Well-founded Phenomena.* Stanford: CSLI Publications.

Belnap, N.D. (1962). Tonk, plonk and plink. *Analysis* **22** (1961/62), 130–134.

Brandom, R.B. (2000). *Articulating Reasons: An Introduction to Inferentialism*, Cambridge Mass.: Harvard University Press.

Dummett, M. (1991). *The Logical Basis of Metaphysics*, London: Duckworth.

Ekman, J. (1998). Propositions in propositional logic provable only by indirect proofs. *Mathematical Logic Quarterly* **44**, 69–91.

Etchemendy, J. (1990). *The Concept of Logical Consequence*, Cambridge Mass.: Harvard University Press.

Fitch. F.B. (1936). A system of formal logic without an analogue to the Curry W operator. *Journal of Symbolic Logic* **1**, 92–100.

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

Gentzen, G. (1934). Untersuchungen über das logische Schließen. *Mathematische Zeitschrift* **39** (1934/35), 176–210, 405–431, English translation ("Investigations into Logical Deduction") in: M.E. Szabo (ed.), *The Collected Papers of Gerhard Gentzen*, Amsterdam: North Holland 1969, 68–131. Quotations are according to Szabo's translation.

Girard, J.-Y. (1971). Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In: J.E. Fenstad (ed.), *Proceedings of the 2nd Scandinavian Logic Symposium (Oslo 1970)*, Amsterdam: North Holland, 63–92.

Groote, P. de (ed.) (1995). *The Curry-Howard Isomorphism.* Louvain-la-Neuve: Academia.

Hallnäs, L. (1991). Partial inductive definitions. *Theoretical Computer Science* **87**, 115–142.

Hallnäs, L. & Schroeder-Heister, P. (1990). A proof-theoretic approach to logic programming. I. Clauses as rules. *Journal of Logic and Computation* **1** (1990/91), 261–283.

Hallnäs, L. & Schroeder-Heister, P. (1991). A proof-theoretic approach to logic programming. II. Programs as definitions. *Journal of Logic and Computation* **1** (1990/91), 635–660.

Hermes, H. (1959). Zum Inversionsprinzip der operativen Logik. In: A. Heyting (ed.), *Constructivity in Mathematics*, Amsterdam: North-Holland, 62–68.

Jäger, G. & Stärk, R.F. (1998). A proof-theoretic framework for logic programming. In: S.R. Buss (ed.), *Handbook of Proof Theory*, Amsterdam: Elsevier, 639–682.

Joachimski, F. & Matthes, R. (2003). Short proofs of normalization for the simply-typed λ-calculus, permutative conversions and Gödels T. *Archive for Mathematical Logic* **42**, 59–87.

Kreuger, P. (1994). Axioms in definitional calculi. In: R. Dyckhoff (ed.), *Extensions of Logic Programming. Proceedings of the 4th International Workshop, ELP '93, St. Andrews, March/April 1993*. Berlin: Springer LNCS, Vol. 798, 196–205.

Lorenzen, P. (1955). Lorenzen, P. *Einführung in die operative Logik und Mathematik*. Berlin: Springer, 2nd ed. 1969.

Martin-Löf, P. (1971). Hauptsatz for the intuitionistic theory of iterated inductive definitions. In: J.E. Fenstad (ed.), *Proceedings of the 2nd Scandinavian Logic Symposium (Oslo 1970)*, Amsterdam: North Holland, 179–216.

Martin-Löf, P. (1984). *Intuitionistic Type Theory*. Napoli: Bibliopolis.

Martin-Löf, P. (1995). Verificationism then and now. In: W. DePauli-Schimanovich et al. (eds.), *The Foundational Debate: Complexity and Constructivity in Mathematics and Physics*, Dordrecht: Kluwer, 187–196.

Martin-Löf, P. (1998). Truth and knowability: On the principles C and K of Michael Dummett. In: H.G. Dales & G. Oliveri (eds.), *Truth in Mathematics*, Oxford: Clarendon Press, 105–114.

Montague, R. (1970). English as a formal language. In: B. Visentini et al. (ed.), *Linguaggi nella Società e nella Tecnica*, Milano. Repr. in: R.H. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, New Haven: Yale University Press 1974, 188–221.

Negri, S. & von Plato, J. (2001a). Sequent calculus in natural deduction style. *Journal of Symbolic Logic* **66**, 1803–1816.

Negri, S. & von Plato, J. (2001b). *Structural Proof Theory*. Cambridge University Press.

Nordström, B., Petersson, K. & Smith, J. (1990). *Programming in Martin-Löf's Type Theory: An Introduction.* Oxford: Clarendon Press.

Plato, J. von (2001). Natural deduction with general elimination rules. *Archive for Mathematical Logic* **40**, 541–567.

Prawitz, D. (1965). *Natural Deduction: A Proof-Theoretical Study.* Stockholm: Almqvist & Wiksell.

Prawitz, D. (1971). Ideas and results in proof theory. In: J.E. Fenstad (ed.), *Proceedings of the 2nd Scandinavian Logic Symposium (Oslo 1970)*, Amsterdam: North Holland, 235–308.

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, 225–250.

Prawitz, D. (1974). On the idea of a general proof theory, *Synthese* **27**, 63–77.

Prawitz, D. (1978). Proofs and the meaning and completeness of the logical constants. In: J. Hintikka et al. (eds.), *Essays on Mathematical and Philosophical Logic*, Dordrecht: Reidel, 25–40 (revised German translation in *Conceptus* **16**, 1982).

Prawitz, D. (1985). Remarks on some approaches to the concept of logical consequence, *Synthese* **62**, 152–171.

Prior, A.N. (1960). The runabout inference ticket. *Analysis* **21** (1960/61), 38–39.

Robering, K. (2001). Ackermann's implication for typefree logic. *Journal of Logic and Computation* **11**, 5–23.

Sambin, G., Battilotti, G. & Faggian, C. (2000). Basic logic: reflection, symmetry, visibility. *Journal of Symbolic Logic* **65**, 979–1013.

Schroeder-Heister, P. (1981). *Untersuchungen zur regellogischen Deutung von Aussagenverknüpfungen.* Ph.D. thesis. University of Bonn 1981. Can be downloaded from the author's homepage.

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., *Computation and Proof Theory. Proceedings of the Logic Colloquium held in Aachen, July 1983, Part II.* Berlin: Springer LNM, Vol. 1104, 399–426.

Schroeder-Heister, P. (1987). *Structural Frameworks with Higher-Level Rules.* Habilitation thesis. Department of Philosophy, University of Constance. Can be downloaded from the author's homepage.

Schroeder-Heister, P. (1991a). Hypothetical reasoning and definitional reflection in logic programming. In: P. Schroeder-Heister (ed.), *Extensions of Logic Program-*

ming. International Workshop, Tübingen, December 1989, Proceedings. Berlin: Springer LNCS, Vol. 475, 327–340.

Schroeder-Heister, P. (1991b). Structural frameworks, substructural logics, and the role of elimination inferences. In: G. Huet & G. Plotkin (eds.), *Logical Frameworks.* Cambridge University Press, 385–403.

Schroeder-Heister, P. (1991c). Uniform proof-theoretic semantics for logical constants. Abstract. *Journal of Symbolic Logic* **56**, 1142.

Schroeder-Heister, P. (1992). Cut-elimination in logics with definitional reflection. In: D. Pearce & H. Wansing (eds.), *Nonclassical Logics and Information Processing. International Workshop, Berlin 1990, Proceedings.* Berlin: Springer LNCS, Vol. 619, 146–171.

Schroeder-Heister, P. (1993). Rules of definitional reflection. In: *8th Annual IEEE Symposium on Logic in Computer Science (Montreal 1993).* Los Alamitos: IEEE Computer Society Press, 222–232.

Schroeder-Heister, P. (1994a). Cut elimination for logics with definitional reflection and restricted initial sequents. *Proceedings of the Post-Conference Workshop of ICLP 1994 on Proof-Theoretic Extensions of Logic Programming (Washington, December 1994).* Can be downloaded from the author's homepage.

Schroeder-Heister, P. (1994b). Definitional reflection and the completion. In: R. Dyckhoff (ed.), *Extensions of Logic Programming. Proceedings of the 4th International Workshop, ELP '93, St. Andrews, March/April 1993.* Berlin: Springer LNCS, Vol. 798, 333–347.

Sommaruga, G. (2000). *History and Philosophy of Constructive Type Theory.* Dordrecht: Kluwer.

Tait, W.W. (1967). Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic* **32**. 198–212.

Tarski, A. (1933). Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica* **1** (1935), 261–405 (translated from the Polish original of 1933, with a postscript). Reprinted in: K. Berka & L. Kreiser (eds.), *Logik-Texte*, Berlin 1971. English translation of the German version in: A. Tarski, *Logic, Semantics, Metamathematics*, Oxford: Clarendon Press 1956.

Tennant, N.W. (1978). *Natural Logic.* Edinburgh University Press.

Troelstra, A.S. & Schwichtenberg, H. (1996). *Basic Proof Theory.* Cambridge University Press, 2nd edition 2000.

Proof-theoretic and constructive consequence

Justifications and reductions

Terminology: I use *"derivation"* for what I before called "derivation structure" ("argument skeleton" in Prawitz), and *"proof"* instead of "argument". So $\mathcal{D}$ denotes a derivation, whereas a proof is a pair $\langle \mathcal{D}, \mathcal{J} \rangle$, also denoted by the Greek letter Π (with and without subscripts). I choose this terminology, as I will have to speak frequently of proofs as arguments and values of functions, which is confusing, if I also use "argument" in a semantical sense. The terminology "derivation" vs. "proof" does not seem to be too bad anyway.

A *proof* is then a pair $\langle \mathcal{D}, \mathcal{J} \rangle$ consisting of a derivation together with a *justification* $\mathcal{J}$, where a justification consists of a set of reductions for derivations. An elementary reduction is a transformation of a derivation whose value has the same end formula and no open assumptions beyond those in the argument (but possibly less). More precisely, as reductions are only required when a derivation is non-canonical, justifications are only defined for non-canonical derivations of the form

$$\frac{\mathcal{D}_1 \ \ldots \ \mathcal{D}_n}{A}\,(X)$$

where (X) is not an introduction inference or an inference in an atomic system S. This means that the transformations within a justification $\mathcal{J}$ are attached to non-canonical rules. It is furthermore required that with each rule not two different reductions are associated. This is expressed by saying that $\mathcal{J}$ is *consistent*. $\mathcal{J}'$ is called a *consistent extension* of $\mathcal{J}$ ($\mathcal{J}' \geq \mathcal{J}$) if $\mathcal{J}'$ is either $\mathcal{J}$ itself or results from $\mathcal{J}$ by adding reductions for rules for which there are no reductions in $\mathcal{J}$, and is furthermore consistent.

It is not necessarily required that the reductions be schematic in the sense that different instances of a rule schema require the same reduction. E.g., it is not excluded that for *modus ponens*

$$\frac{A{\to}B \quad A}{B}$$

the reductions defined differ depending on what formulas A and B stand for. However, in normal circumstances, reductions will be given schematically, as it is the case with the standard reductions of intuitionistic logic. In any case, reductions within justifications have to be interchangeable with substitution, which in the propositional case means the following:

If a reduction j is defined for a derivation $\begin{matrix} A_1 \dots A_n \\ \mathcal{D} \end{matrix}$, such that

$$j\begin{pmatrix} A_1 \dots A_n \\ \mathcal{D} \end{pmatrix} = \begin{matrix} A_1 \dots A_n \\ \mathcal{D}' \end{matrix}, \text{ then for any } \begin{matrix} \mathcal{D}_1 \\ A_1 \end{matrix}, \dots, \begin{matrix} \mathcal{D}_n \\ A_n \end{matrix},$$

$$j\begin{pmatrix} \mathcal{D}_1 & & \mathcal{D}_n \\ A_1 & \dots & A_n \\ & \mathcal{D} & \end{pmatrix} = \begin{matrix} \mathcal{D}_1 & & \mathcal{D}_n \\ A_1 & \dots & A_n \\ & \mathcal{D}' & \end{matrix}.$$

In the quantified case interchange with substitution for individual variables has to be added. Therefore, with respect to the substitution of derivations for open assumptions, reductions are schematic indeed.

Interchangeability with substitution does *not* mean that in

$$\begin{matrix} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \dots A_n \\ \dfrac{\mathcal{D}}{A}\, j_1 \end{matrix},$$

where $A_1, \dots, A_n, A$ are fixed formulas (not schematic letters), j_1 has to be independent of the form of the $\mathcal{D}_i$. In fact, j_1 can vary with varying $\mathcal{D}_1, \dots, \mathcal{D}_n$. However, as soon as one defines some j_2

$$\begin{matrix} A_1 \dots A_n \\ \dfrac{\mathcal{D}}{A}\, j_2 \end{matrix}$$

independently of $\mathcal{D}_1, \dots, \mathcal{D}_n$, then j_1 is determined by j_2, as the argument of j_1 is a substitution instance of the argument of j_2. (In particular, if all reductions in $\mathcal{J}$ are defined for closed derivations only, then no interchangeability requirement is needed.)

An *elementary reduction* associates with a derivation another *derivation*, not a *proof*, at least not in the first place. E.g., in the case of modus ponens, we have the reduction

$$mp_{A\to B}: \quad \dfrac{\begin{matrix} A \\ \mathcal{D} \\ B \end{matrix} \quad \mathcal{D}'}{\begin{matrix} A\to B \quad A \\ B \end{matrix}} \quad \longmapsto \quad \begin{matrix} \mathcal{D}' \\ A \\ \mathcal{D} \\ B \end{matrix}.$$

Although $\begin{matrix}\mathcal{D}'\\ A\end{matrix}$ and $\begin{matrix}A\\ \mathcal{D}\\ B\end{matrix}$ are understood as standing for specific derivations, $mp_{A\to B}$ is independent of the particular forms of $\mathcal{D}$ and $\mathcal{D}'$.

Of course, this *induces* an association between *proofs*, as the reductions associated with steps in $\mathcal{D}$ and $\mathcal{D}'$ are carried over from the argument to the value of $mp_{A\to B}$. However, $mp_{A\to B}$ does not *explicitly refer* to reductions for steps in $\mathcal{D}$ and $\mathcal{D}'$. One might say that reductions for steps in $\mathcal{D}$ and $\mathcal{D}'$ are *parametric* with respect to $mp_{A\to B}$.

This is different with *non-elementary* reductions. Those are reductions which explicitly associate a *proof* rather than just a derivation with a derivation. I.e., the value of a non-elementary reduction may contain new reductions. As an example, consider the reduction j^* whose value contains two elementary reductions, namely $mp_{A\to(B\to C)}$ and $mp_{B\to C}$:

$$j^*\colon\quad \frac{\begin{matrix}\mathcal{D}\\ A\to(B\to C)\end{matrix}}{B\to(A\to C)} \quad\longmapsto\quad \dfrac{\dfrac{\dfrac{\begin{matrix}\mathcal{D}\\ A\to(B\to C)\end{matrix}\qquad \overset{(1)}{[A]}}{B\to C}\, mp_{A\to(B\to C)} \qquad \overset{(2)}{[B]}}{C}\, mp_{B\to C}}{\dfrac{C}{A\to C}\,(1)\text{ I-rule}}}{B\to(A\to C)}\,(2)\text{ I-rule}$$

Again, this induces a reduction going from proofs to proofs, taking reductions associated with steps in $\mathcal{D}$ as parameters. We thus obtain a hierarchy of reductions, starting from the elementary ones. Conversely, given a reduction, it must be possible to reach elementary ones in a finite chain of definitions, as we want to stay predicative.

Apart from the parametric reductions, at present I see no need to also allow for reductions taking proofs (rather than derivations) as *arguments*, i.e., it seems to me that reductions

j: derivation $\longmapsto$ derivation
(elementary case)

j: derivation $\longmapsto$ proof
(non-elementary case)

suffice. (This has to be investigated.) In the following, when we speak of reductions as transformations of *proofs*, this is understood as being parametrically induced by some j in the above sense.

The *full-specification* condition

The example of the reduction j^* given above motivates a certain requirement for justifications which will turn out important in comparison with constructive validity below.

As j^* explicitly refers to the elementary reductions $mp_{A\to(B\to C)}$ and $mp_{B\to C}$, it is not $\{j^*\}$ which justifies the step

$$\frac{A\to(B\to C)}{B\to(A\to C)}\,,$$

but $\{j^*, mp_{A\to(B\to C)}, mp_{B\to C}\}$. In general we should require of a justification $\mathcal{J}$ that it contain *all* reductions, which any reduction in $\mathcal{J}$ refers to. I call this closure condition the *full specification* requirement. A justification should contain all information needed to carry out its reductions, i.e., all reductions apart from the parametric ones.[1] Formally, this condition can be stated as follows: Given a reduction $j \in \mathcal{J}$ of the form

$$j:\ \langle \mathcal{D}_1, \mathcal{J}_1\rangle \quad\longmapsto\quad \langle \mathcal{D}_2, \mathcal{J}_2\rangle$$

Then the full specification requirement demands that $(\mathcal{J}_2 \setminus \mathcal{J}_1) \subseteq \mathcal{J}$, i.e., every reduction beyond those in $\mathcal{J}_1$, which is used to generate $\langle \mathcal{D}_2, \mathcal{J}_2\rangle$, should be in $\mathcal{J}$. In the cases considered here, $\mathcal{J}_1$ would comprise parametric reductions only. In the case of j^*, $\mathcal{J}_2$ would contain the reductions $mp_{A\to(B\to C)}$ and $mp_{B\to C}$, in addition to the parametric reductions in $\mathcal{D}$.

By imposing this requirement, a justification $\mathcal{J}$ of

$$\frac{A_1\ \ldots\ A_n}{A}$$

contains all reductions ever (hereditarily) referred to in spelling out the reduction to be associated with this single step. So all reductions needed are stated in $\mathcal{J}$. We do not allow for reductions to generate further reductions without them being available in $\mathcal{J}$. I consider this to be crucial for the proof-theoretic approach in contradistinction to an approach which may be called *constructive*.

It should be noted that for a derivation $\mathcal{D}$ which is *valid* with respect to $\mathcal{J}$, the full specification requirement is satisfied for $\mathcal{J}$, at least for the subset of $\mathcal{J}$ actually needed to justify $\mathcal{D}$. This is enforced by the specific formulation of the definition of validity. Suppose, $\mathcal{D}$ is non-canonical and closed. Then for $\mathcal{D}$ to be valid with respect to $\mathcal{J}$, $\mathcal{D}$ must reduce with respect to $\mathcal{J}$ to some $\mathcal{D}'$ which is valid with respect to *the very same* $\mathcal{J}$. This means that when reducing $\mathcal{D}$ to $\mathcal{D}'$ using some reduction $j \in \mathcal{J}$, then j cannot refer to a reduction not in $\mathcal{J}$. Similarly, suppose $\begin{array}{c} A_1 \ldots A_n \\ \mathcal{D} \\ A \end{array}$ is valid with respect to $\mathcal{J}$. Then for every $\mathcal{J}' \geq \mathcal{J}$, and for every list of closed derivations $\begin{array}{c} \mathcal{D}_i \\ A_i \end{array}$

[1] If it should turn out that reductions may also refer to non-parametric reductions as arguments, this is of course covered as well.

$(1 \leq i \leq n)$, which are valid with respect to $\mathcal{J}'$, $\begin{array}{c} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \ldots A_n \\ \mathcal{D} \\ A \end{array}$ is valid with respect to *the very same* $\mathcal{J}'$. Again, the reduction $j \in \mathcal{J}'$ which is associated with $\begin{array}{c} A_1 \ldots A_n \\ \mathcal{D} \\ A \end{array}$ (if $\mathcal{D}$ does not end with an introduction step), cannot generate any new reduction, as $\mathcal{J}'$ is not extended.

In this sense, the full specification requirement is built into the definition of validity. Therefore, it might also be called the *full specification feature* of justifications. In any case it seems to me important to state it explicitly, as it contributes to the motivation and understanding of the definition of validity given in comparison with other possible definitions.

Proof-theoretic consequence

Now we may define *proof-theoretic consequence* as follows (I again disregard atomic systems):

A is a proof-theoretic consequence of $A_1, \ldots, A_n$ *with respect to* $\mathcal{J}$ *(*$A_1, \ldots, A_n \models_{\mathcal{J}} A$*), if the one-step derivation*

$$\frac{A_1 \ \ldots \ A_n}{A}$$

is valid with respect to $\mathcal{J}$*, i.e., if for every* $\mathcal{J}' \geq \mathcal{J}$*, and for every list of closed derivations* $\begin{array}{c} \mathcal{D}_i \\ A_i \end{array}$ $(1 \leq i \leq n)$*, which are valid with respect to* $\mathcal{J}'$*,*

(*) $\dfrac{\begin{array}{c} \mathcal{D}_1 \quad \mathcal{D}_n \\ A_1 \ldots A_n \end{array}}{A}$ *is valid with respect to* $\mathcal{J}'$*.*

A is a proof-theoretic consequence of $A_1, \ldots, A_n$ *(*$A_1, \ldots, A_n \models A$*), if* there is a $\mathcal{J}$ *such that A is a proof-theoretic consequence of* $A_1, \ldots, A_n$ *with respect to* $\mathcal{J}$*.*

Note that, analogously to the remark in the previous section, the derivation of the conclusion (*) must be validated with respect to the *very same extension* $\mathcal{J}'$ of $\mathcal{J}$, with respect to which the premiss derivations $\begin{array}{c} \mathcal{D}_i \\ A_i \end{array}$ are validated. $\mathcal{J}'$ cannot be further extended to validate the conclusion. So we are *not* asking that for every $\mathcal{J}' \geq \mathcal{J}$, and

for every list of closed derivations $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$, which are valid with respect to $\mathcal{J}'$, *there is a* $\mathcal{J}'' \geq \mathcal{J}'$ such that (*) is valid with respect to $\mathcal{J}''$. This would lead to a different validity concept with an additional existential quantifier ("*there is* a $\mathcal{J}'' \geq \mathcal{J}'$"), which is more closely related to what might be called constructive rather than proof-theoretic validity, where the full-specification requirement needs not be met.

Proof-theoretic vs. constructive consequence

What is the difference between *proof-theoretic consequence*

$$A_1, \ldots, A_n \models_{\mathcal{J}} A$$

in the sense defined above and *constructive consequence* in the sense that there is a constructive function f transforming valid proofs of $A_1, \ldots, A_n$ into a valid proof of A, formally

$$A_1, \ldots, A_n \models_f A \ .$$

That there might perhaps be no difference at all is suggested by the fact that proof-theoretic consequence is the same as the validity of the one-step derivation

$$\frac{A_1 \ \ldots \ A_n}{A} j$$

where j is a reduction transforming valid closed proofs of $A_1, \ldots, A_n$ into a valid closed proof of A. This procedure j is, of course, a constructive function, and there is no restriction on it in proof-theoretic semantics. So the difference to f can anyway only lie in the $\mathcal{J}$ in which j is embedded.

To answer this question, he have to spell out what "constructive consequence" may mean in our context. Of course, a justification $\mathcal{J}$ as a set of reductions is nothing but a partial constructive function associating proofs with proofs. So what is the special character of $\mathcal{J}$, viewed as a partial constructive function from proofs to proofs, as compared to arbitrary such functions? Let us try to define the validity of proofs of the form $\langle \mathcal{D}, f \rangle$ and see what difference it makes when arbitrary partial functions f rather than partial functions in the sense of $\mathcal{J}$ are considered. The crucial case is the substitution condition in the definition of validity, which also determines logical consequence given through one-step proofs. We again disregard atomic systems S.

In the proof-theoretic case this definition runs as follows:

(P) *An open derivation* $\begin{matrix}A_1 \ldots A_n\\ \mathcal{D}\\ A\end{matrix}$ *, where all open assumptions of* $\mathcal{D}$ *are among* $A_1, \ldots, A_n$*, is valid with respect to* $\mathcal{J}$*, if for every* $\mathcal{J}' \geq \mathcal{J}$*, and for every list of closed*

derivations $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$, *which are valid with respect to* $\mathcal{J}'$, $\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ A\end{matrix}$ *is valid with respect to* $\mathcal{J}'$.

How would we carry this over to the constructive case with partial functions f rather than $\mathcal{J}$? The literal translation for partial functions f with $\mathcal{J}$ replaced by f would be the following:

(C) An open derivation $\begin{matrix}A_1 \ldots A_n\\ \mathcal{D}\\ A\end{matrix}$ *, where all open assumptions of* $\mathcal{D}$ *are among* $A_1, \ldots, A_n$, *is valid with respect to* f, *if for every* $f' \geq f$, *and for every list of closed derivations* $\begin{matrix}\mathcal{D}_i\\ A_i\end{matrix}$ $(1 \leq i \leq n)$, *which are valid with respect to* f', $\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ A\end{matrix}$ *is valid with respect to* f'.

Here $f' \geq f$ means that f' extends f, i.e., f' coincides with f where f is defined. Now in the constructive case one would expect that the form of $\mathcal{D}$ is simply disregarded, i.e. that with closed derivations $\begin{matrix}\mathcal{D}_1\\ A_1\end{matrix}, \ldots, \begin{matrix}\mathcal{D}_n\\ A_n\end{matrix}$ we associate a closed derivation $\begin{matrix}\mathcal{D}'\\ A\end{matrix}$ "out of the blue", without referring to $\mathcal{D}$. Thus f should be construed as an n-place function from $\mathcal{D}_1, \ldots, \mathcal{D}_n$ to $\mathcal{D}'$. However, since we should not exclude in principle that f also depends on $\mathcal{D}$, it would be *more general* to construe f as an $(n+1)$-place function from $\mathcal{D}, \mathcal{D}_1, \ldots, \mathcal{D}_n$ to $\mathcal{D}'$. But then we can as well stick to f as a one-place function of $\begin{matrix}\mathcal{D}_1 \quad \mathcal{D}_n\\ A_1 \ldots A_n\\ \mathcal{D}\\ A\end{matrix}$, as this derivation term contains all the information given by $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and $\mathcal{D}$. This is still not the same as (C), as in (C) just a *single* extension f' of f is being considered, whereas on the constructive approach it would be natural to consider *different* extensions f_i of f for every $\mathcal{D}_i$. However, this is no objection either. Obviously, rather than considering different extensions $f_1, \ldots, f_n$, we might as well consider a single extension f' of f, where f' is defined as the union $f_1 \cup \ldots \cup f_n$ of the f_i, i.e., for arbitrary $\mathcal{D}$, $f'(\mathcal{D}) = f_i(\mathcal{D})$, if for some i $(1 \leq i \leq n)$, f_i is defined for $\mathcal{D}$. This presupposes that the f_i are consistent with each other in that never any two

of them have different values for the same argument, which is only natural to require. This all means that there is no need to deviate from (C) to give (P) a constructive rendering in terms of partial constructive functions.

Obviously, the difference between proof-theoretic and constructive validity does not lie in the difference between (P) and (C), which are the same, but in the different properties of the partial functions $\mathcal{J}$ vs. f being involved. (I tacitly identify $\mathcal{J}$ with the partial function given by the reductions in $\mathcal{J}$.) Apart from the condition that $\mathcal{J}$ interchanges with substitution, which for consistency reasons we would also have to require of f, the only difference seems to be the *full specification* condition. For an arbitrary partial constructive functional f we would allow that as a value it produces a partial function(al) f' which is defined for some arguments for which f is not defined. In our case, for an f we would allow that, if $f(\langle \mathcal{D}_1, f_1\rangle) = \langle \mathcal{D}_2, f_2\rangle$, then f_2 is defined for proofs for which neither f nor f_1 is defined. For example, let f uniformly transform

$$\mathcal{D}_1$$

the proof $\langle \mathcal{D}_1, f_1\rangle$, where $\mathcal{D}_1$ is the derivation $A{\to}(B{\to}C)$, into the proof $\langle \mathcal{D}_2, f_2\rangle$, where $\mathcal{D}_2$ is

$$\dfrac{\dfrac{\dfrac{\mathcal{D}_1 \qquad (1)}{A{\to}(B{\to}C) \quad [A]} \quad (2)}{B{\to}C \qquad\quad [B]}}{\dfrac{C}{A{\to}C}\,(1)}}{B{\to}(A{\to}C)}\,(2)$$

and f_2 contains the *modus ponens* reductions for derivations

$$\dfrac{\mathcal{D}_3 \qquad \mathcal{D}'_3}{A{\to}(B{\to}C) \quad A}{B{\to}C} \quad \text{and} \quad \dfrac{\mathcal{D}_4 \qquad \mathcal{D}'_4}{B{\to}C \quad B}{C}\,.$$

Then f is a constructive justification of the consequence

$$\dfrac{A{\to}(B{\to}C)}{B{\to}(A{\to}C)}$$

which corresponds to to j^*. However, in contradistinction to f, $\{j^*\}$ is not a proof-theoretic justification, as it does not satisfy the full specification condition. Only combined with the *modus ponens* reductions, j^* can serve as a reduction. In constructive semantics we allow for justification functions f, which are functionals, to arbitrarily generate other functions or functionals g with an extended domain, i.e., g may be defined where f was not defined.

Viewed in that way, proof-theoretic semantics is a special form of constructive semantics with the particular emphasis that the justifying functions must be maximally

specified, i.e., all information which might later be invoked, must be given initially in a justification. In other words, the domain of a proof-theoretic justification function $\mathcal{J}$ has to be maximal.

Of course, this can always be achieved given a partial constructive function f on proofs: We just have to hereditarily extend f by defining it for those arguments for which some function in the value of f is defined, which is always possible, as the extensions have to be consistent with the original function. Formally, given a partial constructive justifying function f, we would define such an extension as follows:

$f_0 := f$
$f_{n+1}(\Pi) := f_n(\Pi)$, if f_n is defined for Π
$f_{n+1}(\Pi) := g(\Pi)$, if f_n is not defined for Π and g occurs in the range of f_n, i.e.,
$\quad f_n(\Pi') = \langle \mathcal{D}, g \rangle$ for some Π' and $\mathcal{D}$
f_{n+1} undefined otherwise
$f' := \bigcup f_n$.

Then f' contains all the information some justification $\mathcal{J}$ may contain.

This makes very good sense. Proof-theoretic semantics is not intended to be different from a constructive semantics in that it leads to different results, i.e., by justifying a more limited range of derivations than does constructive semantics. Rather, it just insists that the partial functions used contain maximal information. If a single step

$$\frac{A_1 \ \ldots \ A_n}{A}$$

is to be justified by means of $\mathcal{J}$, then $\mathcal{J}$ contains all information that is needed in the course of the justification, and not just a single reduction for

$$\frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_n \\ A_1 \ldots & A_n\end{array}}{A}$$

which is done in the initial step. In proof-theoretic semantics all information is contained from the very beginning in $\mathcal{J}$ and not hidden in the *value* of some function f for certain arguments.

Afterthought: Term rewriting systems vs. constructive functions

After writing down these notes, it seems to me that there is perhaps a much better way of making the difference between proof-theoretic and constructive consequence explicit. The whole problem arose because a justification $\mathcal{J}$ was from the very beginning

conceived of in the spirit of a partial constructive function, from which the natural question emended, what the exact difference to an *arbitrary* partial constructive function f might be.

Perhaps another framing of $\mathcal{J}$ would be better, namely as a *term rewriting system*. We might expect that a proof-theoretic justification $\mathcal{J}$ should be given in the form of such a system. This would have the following advantages:
1. It would correspond naturally to derivations as term-like structures which can be reduced or rewritten to other derivations.
2. Requirements like interchangeability with substitution are automatically fulfilled due to the term structure.
3. The value of rewriting a term is always a term. There would be no need to consider justifications to occur in values of reductions. This again means that full specification must always be met, as the term rewriting system must provide all possible reduction steps in its definition.

One would then say that proof-theoretic consequence is based on term-rewriting systems as justifications, whereas constructive consequence is based on arbitrary partial functions. Term-rewriting systems can still be read as such functions, but, due to their sophisticated reduction machinery, they have a much higher degree of explicitness. So proof-theoretic semantics would be a specific form of constructive semantics, in which the justifying functions are represented in the form of rewriting systems. The purely functional view simply takes place on a much more abstract level as compared to the concrete term-rewriting (= proof-theoretic) view.

This approach, which has to be investigated in detail, seems to me to be extremely promising.