Structural Frameworks, Substructural Logics, and the Role of Elimination Inferences

Peter Schroeder-Heister*
Universität Tübingen/SNS
Biesingerstr. 10, 7400 Tübingen, Germany
e-mail: schroeder-heister@mailserv.zdv.uni-tuebingen.de

Abstract

Logical inferences are investigated within the context of structural frameworks, in which structural features of deductive systems are separated from logical ones. It is claimed that introduction and elimination inferences operate at different levels. While introduction inferences express logical rules in the genuine sense, elimination inferences are more like structural inferences. This distinction is blurred by certain features of intuitionistic logic, but becomes obvious when systems with restricted structural postulates (substructural logics) are considered.

1 The idea of a structural framework

According to Gentzen [14], in a sequent-style system of logic, we have to distinguish between structural and logical inference schemas. The latter govern the logical content of formulas whereas the former do not refer to logical form. An example of a structural inference schema is that of contraction:

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

an example of a logical inference schema is that of $\vee$-elimination

$$\frac{X \vdash A \vee B \quad Y, A \vdash C \quad Y, B \vdash C}{Y, X \vdash C} \quad ,$$

where A,B and C denote formulas and X and Y denote lists of formulas. This distinction applies to sequent calculi with introductions on both sides of the turnstile (often referred to as "Gentzen-systems"[1]) as well as to sequent-style natural deduction with introduction and elimination inferences for logical constants only on the right side of the turnstile (as

*Draft of a paper which appeared in: G. Huet & G. Plotkin (eds.), Logical Frameworks, Cambridge University Press 1991, pp. 385–403

[1] although Gentzen investigated natural deduction too.

considered in Gentzen [15]). This paper deals mainly with sequent-style natural deduction with a single formula on the right of the turnstile. Only in the final section systems with introductions on the left of the turnstile are considered in connection with computational aspects of deduction. For conceptual simplicity, our investigations are restricted to propositional logic. We also do not deal here with systems with multiple formulas on the right of the turnstile nor with problems of negation.

The idea of "structural frameworks" (see [30]) is twofold: Firstly to extend the notion of structure, and secondly to treat all *logical* content of formulas by means of a database of rules.

Ad 1: The extension of structural means of expression concerns particularly the introduction of a structural implication $\rightarrow$ which is to be distinguished from logical implication $\supset$. It is to supplement structural conjunction which is already present in Gentzen in the form of the comma (to be distinguished from logical conjunction &). Another structural extension (that we are not dealing with in this paper) concerns structural generalization which is to be distinguished from (logical) universal quantification. It is implicit in Gentzen's usage of free variables. This does not mean that at the structural level we are duplicating things which are available at the logical level. Rather, these additional features increase the expressive power of the structural level so as to cover various logical systems in a philosophically plausible and technically uniform way. Apart from that, not every logical constant has a structural analogue (e.g., disjunction has not, at least not in the single-conclusion case). What one would need in a full system is structural conjunction, implication and generalization (of all types), which logically corresponds to a certain fragment of a simple theory of types. In a certain sense, our structural level corresponds to the judgemental level in Martin-Löf's theories.

Ad 2: Instead of considering various logical inference schemas we just consider a single inference schema of rule application. The rules themselves form a database, which is independent of the structural framework itself and may vary. In this way the content-independent inference machinery is separated from the variety of logical systems it may be used to incorporate. One even becomes independent from the logical character of the database: the framework may be used for arbitrary databases of non-logical rules as well, which makes it useful for extensions of logic programming (see [31]). Again, the distinction between inference schemas and rules is a conceptual feature - it is not denied that rules can be replaced by certain inference schemas as in Gentzen.

An example of a structural framework for intuitionistic propositional logic can be given as follows: *Structural atoms* are formulas of propositional logic (letters: A, B, C). *Structural implications* are structural atoms or are of the form $(F \rightarrow G)$ for structural implications F and G (letters: F, G, H). *Structures* are lists of structural implications (letters: X, Y, Z). *Sequents* are of the form $X \vdash F$, *rules* are of the form $X \Rightarrow A$, i.e., sequents have arbitrary structural implications as succedents, whereas rules only have structural atoms as conclusions. *Structural inference schemas* are the following:

$$(Reflexivity)\ \frac{}{F\vdash F} \qquad (Permutation)\ \frac{X,F,G,Y\vdash H}{X,G,F,Y\vdash H}$$

$$(Contraction)\ \frac{X,F,F,Y\vdash H}{X,F,Y\vdash H} \qquad (Thinning)\ \frac{X\vdash H}{X,F\vdash H}$$

$$(\to I)\ \frac{X,F\vdash G}{X\vdash F\to G} \qquad (\to E)\ \frac{X\vdash F \quad Y\vdash F\to G}{Y,X\vdash G}$$

$$(\vdash\Rightarrow)\ \frac{X_1\vdash F_1 \quad \ldots \quad X_n\vdash F_n}{X_1,\ldots,X_n\vdash A}\ \text{if } F_1,\ldots,F_n\Rightarrow A \text{ is in the database}$$

The schema $(\vdash\Rightarrow)$ also covers the limiting case where the database rule has no premisses and the upper sequents of the inference are lacking.

The database may contain logical rules such as

$$A, B\Rightarrow A\&B \qquad A\to B\Rightarrow A\supset B$$

$$A\Rightarrow A\vee B \qquad B\Rightarrow A\vee B$$

$$\vdots$$

However, in principle this approach is not confined to logic; the database may contain extra-logical rules also. For logical elimination inferences see §3 below.

In contrast to previous descriptions based on "higher-level rules" (see [29, 30]), this paper strictly distinguishes between structural implications $F\to G$, which are obtained by iterating $\to$ to the left and to the right, and rules $X\Rightarrow A$, which have atoms as conclusions. The concept of "higher-level rules" mixed these two things up by using $\Rightarrow$ both as a sign for structural implication and as the rule arrow. It is essential for a computational interpretation of database rules that they may contain structural implications as premisses, but themselves are different from structural implications. Rules have a normative (definitional) meaning governed by $(\vdash\Rightarrow)$ (and by the dual principle (IP) discussed in §4). They *define* the "world" one is dealing with. Structural implications have a declarative (descriptive) meaning governed by $(\to I)$ and $(\to E)$. They make assumptions or assertions *about* this world (see [28] for further general remarks on that topic).

Our approach gives a natural view of logic programming, if one takes atomic formulas as atoms and considers database rules $F_1,\ldots,F_n\Rightarrow A$ as program clauses. Such a logic programming language permits structural implications in the bodies of clauses and thus extends definite Horn clause programming (see [20]).

2 Structural frameworks for substructural logics

Substructural logics are logical systems in which certain structural principles which are standard in the intuitionistic case are not available.[2] Examples are BCK-logic, in which

[2]The term "substructural" was proposed by Kosta Došen.

Contraction is lacking, relevant logics, in which Thinning is dropped, linear logic, which has neither Contraction nor Thinning, or the Lambek calculus which is without Permutation and which in one of its versions does not even have associativity of antecedent formulas. The interest in such systems is increasing, partly due to applications of these logics in computer science and theoretical linguistics. For an overview see [11].

Structural frameworks corresponding to substructural logics can be developed by imposing the restrictions mentioned on the structural inference schemas and by extending the notion of a structural implication so as to incorporate these modifications. The logical principles themselves would, as before, be formulated as database rules. In what follows, frameworks for some prominent logics will be considered as examples, without any claim to completeness: the associative Lambek system and relevant systems without distribution as pure single-family systems, relevant logic with distribution as a pure two-family-system and BCK-logic and linear logic as single-family systems which in addition allow for structure-independent rule-application. Following Belnap's [3] classification, single-family and two-family-systems are based on one or two, respectively, structural associations in the antecedent of a sequent. In the single-family case, we denote it, as usual, by the comma, which is understood as a multi-ary structural connective. In the two-family case we use the semicolon for the second structural connective. By "structure-independent" rule application we mean an inference schema for rule application which (unlike $(\vdash\Rightarrow)$) does not refer to a particular structural association within the antecedent of a sequent (this will become clear from the examples below). Belnap speaks of "families", since in his framework there are, for each structural association, corresponding logical conjunctions, disjunctions, negations, implications and constants. We borrow this terminology, but basically consider conjunctions and implications only. For the systems we are interested in, we need not treat the comma or the semicolon as a binary structural connective.

2.1 Pure single-family-systems: Relevant logic without distribution, Lambek calculus

Here we consider frameworks where, as in the intuitionistic case, there is exactly one way of associating structural implications to form a structure, which is denoted by the comma. A structural framework for relevant logic without distribution results from the framework for intuitionistic logic by taking away the postulate of Thinning; i.e., Permutation and Contraction are still available. So, as before antecedents of sequents may be viewed as finite sets. This system of relevant logic is investigated among other single-family systems in [8]. It is not the system of the mainstream of the relevant logic tradition, which rather favours systems in which the distribution law $A\&(B\vee C)\vdash(A\&B)\vee C$ holds. This distribution law cannot be obtained by just dropping Thinning in the structural framework. The proof-theoretically most elegant way is to consider a two-family structural framework (see §2.3 below).

In the Lambek calculus as first described in [22] none of the structural postulates of Permutation, Contraction and Thinning is available. This means, we have associativity of antecedent formulas, i.e., we can write structures as lists, but do not have any further structural postulate in the traditional sense apart from Reflexivity. In the sense of our structural frameworks we do have, of course, further postulates, namely those governing

structural implications and the application of rules.[3] Due to the fact that Permutation is lacking, two structural implications are available, which are here denoted by $\stackrel{r}{\rightarrow}$ and $\stackrel{l}{\rightarrow}$, i.e., non-atomic structural implications may be of the form $F \stackrel{r}{\rightarrow} G$ or $F \stackrel{l}{\rightarrow} G$. The structural inference schemas governing $\stackrel{r}{\rightarrow}$ and $\stackrel{l}{\rightarrow}$ are the following:

$$(\stackrel{r}{\rightarrow}I)\ \frac{X, F \vdash G}{X \vdash F \stackrel{r}{\rightarrow} G} \qquad (\stackrel{r}{\rightarrow}E)\ \frac{X \vdash F \quad Y \vdash F \stackrel{r}{\rightarrow} G}{Y, X \vdash G}$$

$$(\stackrel{l}{\rightarrow}I)\ \frac{F, X \vdash G}{X \vdash F \stackrel{l}{\rightarrow} G} \qquad (\stackrel{l}{\rightarrow}E)\ \frac{X \vdash F \quad Y \vdash F \stackrel{l}{\rightarrow} G}{X, Y \vdash G}.$$

Based on these two structural implications, it is possible to include in the database introduction rules for Lambek's two logical implications / and \:

$$A \stackrel{r}{\rightarrow} B \Rightarrow B/A \qquad A \stackrel{l}{\rightarrow} B \Rightarrow A \backslash B\ .$$

It is obvious that the postulates of the structural framework are essential for which logical constants can be defined in the database. So it is not only the database of logical rules which makes the difference between logics, but the structural apparatus governing this database. This gives rise to criteria of logicality according to which one may distinguish between mere extensions of logical systems obtained by adding rules for further constants to an existing database and rival systems obtained by changing the structural postulates for an existing system.[4]

2.2 Pure two-family-systems: Relevant logic with distribution

In order to obtain a Gentzen-type system for relevant logics in which the distribution law $A\&(B \vee C) \vdash (A\&B) \vee C$ holds, Dunn [13] proposed to use two structural conjunctions for associating antecedent formulas: an extensional one denoted by the comma and an intensional one denoted by the semicolon, with Thinning holding for the comma but not for the semicolon.[5] To develop a structural framework based on this idea, we first introduce an additional operator $\dot{\rightarrow}$ for forming structural implications. Then we distinguish between extensional and intensional structures. Every structural implication is both an extensional and an intensional structure. *Extensional structures* are of the form $(X_1, \ldots, X_n)$ for intensional structures $X_1, \ldots, X_n$, whereas *intensional structures* are of the form $(X_1; \ldots; X_n)$ for extensional structures $X_1, \ldots, X_n$. Furthermore, for $n = 0$ we have the empty extensional structure $\emptyset_e$ and the empty intensional structure $\emptyset_i$ as limiting cases. A *structure*

[3] In the even weaker non-associative Lambek calculus introduced in [21] one would have to treat the comma as a binary operator, as done in the uniform treatment of various systems in [3]. Despite the fact that Contraction is lacking, we treat the Lambek calculus as a single-family system as in the original source [22]. One may of course add structure-independent rule application, yielding a system of the type of §2.3.

[4] This idea is closely related to Došen's [6] view of logical constants as punctuation marks, although Došen's structural apparatus is somewhat different from ours.

[5] For later applications and philosophical discussions of this approach see [3, 5, 16, 24, 26, 32].

is an extensional or an intensional structure. So extensional and intensional associations may be nested within a structure. Sequents are of the form $X \vdash F$ for structures X and structural implications F. Rules are either of the form $F_1, \ldots, F_n \Rightarrow A$ (extensional rules) or of the form $F_1; \ldots; F_n \Rightarrow A$ (intensional rules) for structural implications F_i and structural atoms A with limiting cases $\emptyset_e \Rightarrow A$ and $\emptyset_i \Rightarrow A$. When we use a notation like $X, F, (Y; Z) \vdash G$ for a sequent, the commas and semicolons in the antecedent denote either concatenation of equal-type structures or structure-forming operations, whichever is applicable.[6] Then the structural inference schemas include all schemas of the intuitionistic case as given in §1 (but now with X, F and G read as structures and structural implications in the present sense), and furthermore

$$(Permutation;)\ \frac{X; F; G; Y \vdash H}{X; G; F; Y \vdash H} \qquad (Contraction;)\ \frac{X; F; F; Y \vdash H}{X; F; Y \vdash H}$$

$$(\dot{\rightarrow} I)\ \frac{X; F \vdash G}{X \vdash F \dot{\rightarrow} G} \qquad (\dot{\rightarrow} E)\ \frac{X \vdash F \quad Y \vdash F \dot{\rightarrow} G}{Y; X \vdash G}$$

$$(\vdash \Rightarrow;)\ \frac{X_1 \vdash F_1 \quad \ldots \quad X_n \vdash F_n}{X_1; \ldots; X_n \vdash A} \text{ if } F_1; \ldots; F_n \Rightarrow A \text{ is in the database}$$

Here the limiting cases for $n = 0$ of $(\vdash \Rightarrow)$ and $(\vdash \Rightarrow;)$ permit to infer $\emptyset_e \vdash A$ or $\emptyset_i \vdash A$, depending on whether an extensional or an intensional rule is applied.

This means that that we have corresponding principles for the comma with the associated structural implication $\rightarrow$ and for the semicolon with the associated structural implication $\dot{\rightarrow}$, with the exception that for the comma, but not for the semicolon, Thinning is available.

Introduction rules for logical constants of relevant logics may be the following database rules

$$A, B \Rightarrow A \& B \qquad A \rightarrow B \Rightarrow A \supset B \qquad \emptyset_e \Rightarrow t_e$$

$$A; B \Rightarrow A \circ B \qquad A \dot{\rightarrow} B \Rightarrow A \texttt{->} B \qquad \emptyset_i \Rightarrow t_i$$

to define extensional conjunction (&), implication ($\supset$) and truth (t_e) as well as intensional conjunction (fusion, $\circ$), intensional (relevant) implication (->) and intensional truth (t_i).

[6] Therefore, if X is an extensional structure, the first and the second comma are appending F and $(Y; Z)$ to X. If X is an intensional structure, these commas form a three-element extensional structure out of X, F and $(Y; Z)$. Similarly, if Y and Z are both intensional structures, the semicolon appends the elements of Z to Y. If Y is intensional and Z extensional, the semicolon appends the extensional structure Z to Y. If both are extensional the semicolon forms a two-element intensional structure, etc. If X is an extensional structure, then $X, \emptyset_e$ is the same as X, otherwise the empty extensional structure $\emptyset_e$ is appended to X, and likewise for $X, \emptyset_i$ or $X; \emptyset_e$ or $X; \emptyset_i$.

2.3 Systems with structure-independent rule application: BCK-logic, linear logic

In BCK-logic (see [25]) and linear logic (see [17][7]) one considers a single structure-sensitive association as in §2.1, but in addition a structure-independent schema for and-introduction:

$$\frac{X\vdash A \quad X\vdash B}{X\vdash A\wedge B}.$$[8]

In our context this corresponds to a general schema of structure-independent rule application. Besides rules $F_1,\ldots,F_n\Rightarrow A$, which are applied according to the schema $(\vdash\Rightarrow)$, we consider rules of the form $F_1,,\ldots,,F_n\Rightarrow A$, with premisses divided by double commas, which are applied as follows:

$$(\vdash\Rightarrow,,)\ \ \frac{X\vdash F_1 \quad \ldots \quad X\vdash F_n}{X\vdash A}\ \text{ if } F_1,,\ldots,,F_n\Rightarrow A \text{ is in the database.}$$

In a structural framework for BCK-logic we have the inference schemas of Reflexivity, Permutation, Thinning, $(\rightarrow I)$, $(\rightarrow E)$, $(\vdash\Rightarrow)$ and $(\vdash\Rightarrow,,)$, i.e., Contraction is lacking as compared to the intuitionistic case. In a structural framework appropriate for linear logic Thinning is excluded in addition. In the database we can now include introduction rules for two conjunctions:

$$A, B\Rightarrow A\&B \qquad\qquad A,, B\Rightarrow A\wedge B$$

As for the systems considered before, the structural framework itself is independent of what kind of rules are available in the database. There is no need for them to be logical rules.

Mixtures of $(\vdash\Rightarrow)$ and $(\vdash\Rightarrow,,)$ are also possible, e.g., by permitting rules such as $F_1,(F_2,,F_3)\Rightarrow A$ which would be applied as follows:

$$\frac{X\vdash F_1 \quad Y\vdash F_2 \quad Y\vdash F_3}{X,Y\vdash A}.$$

To avoid a possible misunderstanding: The distinction between structure-sensitive and structure-independent rule application is no particular feature of weak substructural logics, but exists for intuitionistic logic too. However, in the presence of Contraction and Thinning, these cases can be mutually reduced to each other.

[7] By "linear logic" we mean what Girard calls "rudimentary linear logic", i.e., we are just considering the propositional fragment without exponentials. For a treatment of the structural features of linear logic within the framework of display logic see [4].

[8] By "structure-independent" as opposed to "structure-sensitive" we mean that the lower sequent of an inference schema does not refer to a particular way of structuring the antecedent as does a schema like

$$\frac{X\vdash A \quad Y\vdash B}{X,Y\vdash A\&B} \quad\text{or}\quad \frac{X\vdash A \quad Y\vdash B}{X;Y\vdash A\circ B}.$$

For the case of conjunction, Girard's terminology is "additive" versus "multiplicative" ([17]) or "alternative" versus "cumulative" ([18], Appendix B).

One may, of course, add structure-independent rule application to a two-family system, yielding, e.g., a relevant logic with conjunctions &, $\circ$ and $\wedge$. More generally, one may develop the idea of a universal structural framework that would allow to make all the distinctions mentioned so far. Such a meta-framework would have to provide facilities for defining structural postulates themselves in a uniform way. Došen's [9] idea of sequents of higher levels and of horizontalizing inference schemas into such sequents may become fruitful in that respect. This approach might also lead to a conceptually perspicuous treatment of modal logics within structural frameworks. The idea of such a general structural framework goes in a different direction than do most logical frameworks, since it comprises a whole world of systems even at the propositional level. Present-day logical frameworks normally start with problems of higher types and quantification and take the structural postulates of intuitionistic logic for granted.

3 The problem of elimination rules

So far the examples of logical rules have been only introduction rules. In the structural framework considered in §1 with unrestricted structural postulates, elimination rules can be added to the database as well. The $\vee$-elimination rule would read

$$(\vee E\ rule_i) \quad A \vee B, (A \rightarrow C), (B \rightarrow C) \Rightarrow C$$

(the index i here stands for "intuitionistic"). Following the pattern of the $\vee$-elimination rule, in [29] the following general schema for introduction and elimination rules for an n-ary sentential operator s was proposed:

$$(sI\ rules) \quad X_1 \Rightarrow s(A_1, \ldots, A_n) \quad \ldots \quad X_n \Rightarrow s(A_1, \ldots, A_n)$$

$$(sE\ rule) \quad s(A_1, \ldots, A_n), (X_1 \rightarrow C), \ldots, (X_n \rightarrow C) \Rightarrow C\ .$$

Here the X_i stand for structures (lists of structural implications), in which only formulas built up from $A_1, \ldots, A_n$ by means of logical operators already defined occur. For example, in the case of s being implication, we have $n = 1$ and $X_1 = (A_1 \rightarrow A_2)$; in the case of disjunction we have $n = 2$ and $X_1 = A_1$, $X_2 = A_2$; in the case of equivalence we have $n = 1$ and $X_1 = ((A_1 \rightarrow A_2), (A_2 \rightarrow A_1))$, etc. It can be shown that in all cases $(sE\ rule)$ is equivalent to the "standard" elimination rule. For example,

$$(\supset E) \quad A \supset B, ((A \rightarrow B) \rightarrow C) \Rightarrow C$$

is equivalent to *modus ponens*

$$A \supset B, A \Rightarrow B\ .$$

According to this view, both introduction and elimination rules are part of the database of logical rules, the elimination rules being generated in a uniform way from the introduction rules. However, this becomes problematic when substructural logics are considered. The idea that the general schema $(sE\ rule)$ for elimination rules can be taken to be the same in all structural frameworks (see [30]) turns out to be infeasible.

We take the $\vee$-elimination rule ($\vee E\ rule_i$) as an example. Applying this rule is equivalent to using the following inference schema:

$$(\vee E\ schema_i) \quad \frac{X \vdash A \vee B \qquad Y, A \vdash C \qquad Z, B \vdash C}{X, Y, Z \vdash C}.$$

In the intuitionistic structural framework this is equivalent to

$$(\vee E\ schema) \quad \frac{X \vdash A \vee B \qquad Y, A \vdash C \qquad Y, B \vdash C}{Y, X \vdash C}.$$

However, in the absence of Thinning and Contraction these two schemas are not equivalent, and ($\vee E\ schema$) is more appropriate than ($\vee E\ schema_i$), since otherwise desirable results cannot be obtained. In systems without Thinning ($\vee E\ schema$) permits normalization of proofs whereas ($\vee E\ schema_i$) does not. To see that, suppose $X \vdash A \vee B$ is inferred from $X \vdash A$, and ($\vee E\ schema_i$) is applied in the next step. Then the standard contraction of this redex yields $Y, X \vdash C$ (the conclusion of ($\vee E\ schema$)) rather than $X, Y, Z \vdash C$ (for which Thinning would be needed). In a system without Contraction replacement of equivalent formulas becomes problematic with ($\vee E\ schema_i$). For example, suppose that $Y, A \vee B \vdash C$ and $A \dashv\vdash A'$ have been proved. Then $Y, A' \vdash C$ and $Y, B \vdash C$ can be inferred, from which by ($\vee E\ schema_i$) $Y, Y, A' \vee B \vdash C$ rather than $Y, A' \vee B \vdash C$ is obtained.[9].

However, this does not necessarily speak against elimination inferences as rules. By using the mixture of structure-sensitive and structure-independent rule application mentioned in §2.3 one may obtain the effect of ($\vee E\ schema$) by means of the rule

$$(\vee E\ rule) \quad A \vee B, ((A \to C) ,, (B \to C)) \Rightarrow C\ .$$

In all systems where one can treat structures as multisets, i.e., where one has Permutation, ($\vee E\ rule$) would be appropriate.

However, in a Lambek-style system, where Permutation is lacking, or in a two-family system with the possibility of a nested structuring of antecedents, the situation becomes different. In both cases the $\vee$-elimination schema has to be formulated as

$$(\vee E\ schema') \quad \frac{X \vdash A \vee B \qquad Y[A] \vdash C \qquad Y[B] \vdash C}{Y[X] \vdash C}\ ,$$

where the square brackets denote the occurrence of a certain formula within a structure. This includes ($\vee E\ schema$) as a special case. In a Lambek-style framework, due to the fact that Permutation is lacking, $Y[A] \vdash C$ does not necessarily imply $Y, A \vdash C$, so that ($\vee E\ schema$) is strictly weaker than ($\vee E\ schema'$). This again means that ($\vee E\ schema'$) cannot be expressed by the database rule ($\vee E\ rule$) (with $\to$ understood as $\stackrel{r}{\to}$ or $\stackrel{l}{\to}$). A similar argument applies to two-family-systems, since an A in a context $Y[A]$ cannot necessarily be extracted and moved out of the context. For example, a sequent $(A_1; A), A_2 \vdash C$ is not normally equivalent to some $X, A \vdash B$ or $X; A \vdash B$.

[9]In particular, without Contraction, ($\vee E\ schema_i$) does not guarantee the full uniqueness of disjunction in the sense that $X, A \vee B \vdash C$ is always equivalent to $X, A \vee^* B \vdash C$, with $\vee^*$ being a duplicate of $\vee$ with the same rules. Uniqueness is dual in a way to conservativeness, which again is related to normalization (see [10, 12]).

There seems to be a way out of this problem, namely by allowing structural implications and not just structural atoms to be conclusions (heads) of rules. The $\vee$-elimination rule could then be formulated as

$$(\vee E\ rule') \quad A \vee B, ((A \rightarrow F),, (B \rightarrow F)) \Rightarrow F\ ,$$

which has the effect of $(\vee E\ schema')$. For example, in the two-family system of §2.3, $(A_1; A), A_2 \vdash C$ is equivalent to $A \vdash A_1 \dot{\rightarrow} (A_2 \rightarrow C)$. So from $X \vdash A \vee B$ and $(A_1; A), A_2 \vdash C$ and $(A_1; B), A_2 \vdash C$ we obtain $X \vdash A_1 \dot{\rightarrow} (A_2 \rightarrow C)$ by applying $(\vee E\ rule')$, which is equivalent to $(A_1; X), A_2 \vdash C$. A similar example can be given for the Lambek-style system: suppose we have $X \vdash A \vee B$ and $A_1, A, A_2 \vdash C$ and $A_1, B, A_2 \vdash C$. Then we obtain $A \vdash A_1 \stackrel{l}{\rightarrow} (A_2 \stackrel{r}{\rightarrow} C)$ and $B \vdash A_1 \stackrel{l}{\rightarrow} (A_2 \stackrel{r}{\rightarrow} C)$ and thus $X \vdash A_1 \stackrel{l}{\rightarrow} (A_2 \stackrel{r}{\rightarrow} C)$ by applying $(\vee E\ rule')$, which is equivalent to $A_1, X, A_2 \vdash C$.

In the single-conclusion case, this approach (which was followed in [30]) has a similar effect as Belnap's [3] use of an involutive structural negation in a multiple-conclusion framework, by means of which formulas can be moved arbitrarily between the two sides of a sequent. However, this runs counter to the intuitions of structural frameworks, which are to separate the structural aspects of a system from its logical (database) content. Structural implication(s) should only be governed by structural introduction and elimination principles like $(\rightarrow I)$ and $(\rightarrow E)$ and not by rules in the database. Structural implications in the heads of rules would put material (non-structural) content into structural implications, making structural implications logical entities.

The fact that $(\vee E\ schema')$ and not $(\vee E\ schema)$ is the correct representation of $\vee$-elimination reveals in general, quite independently of our notion of a structural framework, a deep asymmetry betwen introduction and elimination inferences, more precisely, a difference in the notion of discharging assumptions. Whereas in $(\vee E\ schema')$, one has to consider arbitrary embeddings of formulas in structures (denoted by square brackets), this is not the case with introduction schemas. For example, in the Lambek calculus the schema for $\backslash$-introduction has to be formulated as

$$\frac{A, X \vdash B}{X \vdash A \backslash B}$$

with A at a specified (namely leftmost) place on the left side of the turnstile and *not* as

$$\frac{X[A] \vdash B}{X \vdash A \backslash B}$$

with A arbitrarily embedded in a context. Otherwise, under certain natural conditions, the structural law of Permutation could be proved by applying principles for logical implications.[10]

This is blurred in the intuitionistic system with its simple structural assumptions and also in weaker systems which can be based on multisets as assumption-structures. From

[10] This asymmetry between introduction and elimination rules will be treated in more detail in [27]. It also applies to sequent systems with more than one formula in the succedent. The failure of conservativeness of the wrong implication law over the structural law of Permutation was observed by Wansing [33].

the standpoint of a general structural framework, such structural assumptions are just limiting cases and cannot be taken as universal: two-family systems and systems without Permutation cannot be excluded without giving up the conceptual unity of structural frameworks with different structural assumptions. Conclusions to be drawn about structural frameworks in general therefore also apply to the intuitionistic case. This means that elimination-inferences cannot be properly treated as database rules but have to be given a different role.

4 Elimination inferences as structural inferences

If elimination inferences cannot be incorporated into the database, how else can they be treated? To formulate them in the ordinary way as inference schemas like ($\vee E$ *schema'*) and add them to the structural inference schemas would destroy the idea of a structural framework, in which the structural part is kept apart from the content (logical or other) which is put into the database.

A way out is indicated by the fact that the elimination inferences follow a general pattern. This general pattern can be formulated as

$$(sE\ schema)\quad \frac{X\vdash s(a_1,\ldots,a_n)\qquad Y[X_1]\vdash C\quad\ldots\quad Y[X_n]\vdash C}{Y[X]\vdash C}\ ,$$

if the introduction rules for s are given by (sI *rules*). Unlike (sE *rule*), which turned out to be problematic with restricted structural postulates (at least with two-family systems and systems without Permutation), (sE *schema*) would be appropriate for all structural frameworks without structure-independent sI-rules. So we are not putting specific logical inference schemas into a structural framework, but one general schema. By further generalizing it we can even make it entirely independent of the specific form of rules for logical constants. In this way we obtain a general structural principle which applies to any database of rules, databases of logical introduction rules just being a special case.

More precisely, we formulate a general elimination or inversion principle for arbitrary structural atoms, not only for logically compound formulas. Given a database of rules, define

$$\mathbf{D}(A) := \{Z : Z\Rightarrow A \text{ is a substitution instance of a database rule}\}.$$

Then the general elimination schema is formulated as

$$(IP)\quad \frac{X\vdash A\qquad \{Y[Z]\vdash C : Z\in\mathbf{D}(A)\}}{Y[X]\vdash C}\ .$$

It is obvious that in the case of logical constants with (sI *rules*) as introduction rules, (sE *schema*) is a special case of (IP). However, (IP) is general enough to be counted as a structural postulate as opposed to database rules which govern contents. This does not mean that (IP) has to be an ingredient of any structural framework. As with every structural postulate, one may discuss what happens if it is present and if it is lacking, obtaining various systems that way. In the case of logic, it represents a way of dealing uniformly with elimination inferences for systems weaker than intuitionistic logic. In the

case of other databases, it represents a way of treating arbitrary rules as introduction rules for atoms, allowing to invert these rules in a certain way. This is why we call (IP) an "inversion principle"[11].

It may be noted that (IP) becomes an infinitary schema if $\mathbf{D}(A)$ is infinite. However, natural restrictions can be formulated that make $\mathbf{D}(A)$ finite when (IP) is applicable.[12] In the case of logical constants, $\mathbf{D}(A)$ is finite due to the restrictions on introduction rules (such as the one that the premisses X_i in an introduction rule $X_i \Rightarrow s(A_1, \ldots, A_n)$ should contain no formulas except $A_1, \ldots, A_n$).

When structure-independent rule application is available, as in frameworks suitable for linear or BCK logic, $\mathbf{D}(A)$ and (IP) have to be formulated in a more general way. For example, take the $\wedge$-introduction rule $A,, B \Rightarrow A \wedge B$. The elimination inferences should comprise the two schemas

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

rather than the single one

$$\frac{X \vdash A \wedge B \qquad Y[A, B] \vdash C}{Y[X] \vdash C} \ ,$$

which would be appropriate for the structure-sensitive conjunction &. In general, let U be of the form $(F_1, \ldots, F_n)$ or $(F_1,, \ldots,, F_n)$. Then U is called a *premiss structure* (which is not a structure in the genuine sense, since the double comma is not considered a structure-forming operation that can occur on the left side of $\vdash$). A *selection function* is a function f operating on premiss-structures such that

$$\begin{array}{l} f(U) = U \text{ if } U = (F_1, \ldots, F_n) \\ f(U) = F_i \text{ for some } i \ (1 \leq i \leq n) \text{ if } U = (F_1,, \ldots,, F_n) \ . \end{array}$$

Then we define for each selection function f

$$\mathbf{D}^f(A) := \{f(U) : U \Rightarrow A \text{ is a substitution instance of a database rule}\}.$$

The generalized inversion principle consists of a set of schemas, one for each selection function:

$$(IP_{gen}) \quad \left\{ \frac{X \vdash A \qquad \{Y[Z] \vdash C : Z \in \mathbf{D}^f(A)\}}{Y[X] \vdash C} \ : f \text{ selection function} \right\}.$$

Obviously, (IP) is a limiting case of (IP_{gen}), if $\mathbf{D}^f(A)$ is the same for all f, which holds if there is no structure-independent rule by means of which A can be obtained.

Computationally, the schema (IP) is not well tractable since the A in the left premiss does not occur in the conclusion. For computational purposes a sequent-style system

[11] In fact, (IP) is closely related to Lorenzen's "inversion principle" (see [23]), a relationship which cannot be spelled out here.

[12] One basically requires that $(\mathbf{D}(A))\sigma = \mathbf{D}(A\sigma)$ for any substitution σ, and therefore that (IP) is closed under substitution. For further discussion see [20]).

with introductions on the left side of the turnstile is more appropriate. To obtain such a system, we replace in the structural frameworks considered the Reflexivity principle by

$$\frac{}{A \vdash A}$$

(which refers to structural atoms rather than arbitrary structural implications) and the $(\rightarrow E)$ principle by

$$(\rightarrow\vdash) \quad \frac{X \vdash F \qquad Z, G \vdash H}{Z, (F \rightarrow G), X \vdash H} \ .$$

Analogous changes apply to cases where several structural implications are available. The schema (IP) is then reformulated as a schema for the introduction of an atom A on the left side of the turnstile:

$$(\Rightarrow\vdash) \quad \frac{\{Y[Z] \vdash C : Z \in \mathbf{D}(A)\}}{Y[A] \vdash C} \ ,$$

which in a certain sense is dual to $(\vdash\Rightarrow)$. (Here the letter C may equivalently be replaced by F, which gives more computational flexibility.) This is exactly the inference schema which is discussed in the context of logic programming in [19], [20] and [28], upon which the programming language GCLA is based ([1, 2]). In that context it was called the "**D**-rule" or the rule of "local reflection" and was motivated from an entirely different point of view which had nothing to do with structural postulates and restrictions thereof but with considerations concerning inductive definitions and treating databases in that way. The fact that a different approach leads to exactly the same principle confirms its conceptual significance.

The formulation of (IP) as a computationally meaningful principle for the introduction of atoms on the left side is another argument against the treatment of elimination inferences as database rules, even if this is possible as it is in the intuitionistic case. The structural atom C in these rules, which unifies with every other structural atom, excludes any computational reading. When applied according to $(\vdash\Rightarrow)$, database rules produce only right-introduction inferences, whereas elimination inferences receive computational significance by formulating them as left-introduction inferences. The schema $(\Rightarrow\vdash)$ carries out the idea of computationally relevant introductions of (structural) atoms on the left side of the turnstile in a uniform manner, independently of their specific logical form (if they have one at all).

The generalized version of $(\Rightarrow\vdash)$ corresponding to (IP_{gen}) is the following:

$$(\Rightarrow\vdash_{gen}) \quad \left\{ \frac{\{Y[Z] \vdash C : Z \in \mathbf{D}^f(A)\}}{Y[A] \vdash C} \ : \ f \text{ selection function} \right\} \ .$$

It might be mentioned that for logic programming the combinatorial complexity of $(\Rightarrow\vdash_{gen})$ may lead to difficulties, although database rules are most easily understood and handled as structure-independent rules according to $(\vdash\Rightarrow,,)$ of §2.3. This is no problem for intuitionistic logic, where Thinning and Contraction is available, but becomes problematic when evaluation procedures for weaker systems are considered (see [31]).

The view we have arrived at is that of a structural framework which, besides the handling of structural association (of premisses of rules or antecedents of sequents) and of structural implication, may include inversion principles. In the case of logically compound formulas, these inversion principles instantiate to elimination inferences. The (logical or extra-logical) content of formulas is expressed in a database of rules which, in the case of logical composition, are introduction rules in the ordinary sense.

Acknowledgements. The research reported here was supported by the Fritz Thyssen Foundation. I would like to thank Kosta Došen, Venkat Ajjanagadde and two anonymous referees for helpful comments and suggestions.

References

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

[2] Aronsson, M., Eriksson, L.-H., Hallnäs, L. & Kreuger, P. A survey of GCLA: A definitional approach to logic programming. In: P. Schroeder-Heister (ed.), *Extensions of Logic Programming. International Workshop, Tübingen, FRG, December 1989, Proceedings.* Springer LNCS, Vol. 475, Berlin (1991), 49–99.

[3] Belnap, N. D. Display logic. *Journal of Philosophical Logic*, **11** (1982), 375–417.

[4] Belnap, N. D. Linear logic displayed. *Notre Dame Journal of Formal Logic*, **31** (1990), 14–25.

[5] Belnap, N. D., Gupta, A. & Dunn, J. M. A consecution calculus for positive implication with necessity. *Journal of Philosophical Logic*, **9** (1980), 343–362.

[6] Došen, K. Logical constants as punctuation marks. *Notre Dame Journal of Formal Logic*, **30** (1989), 362–381.

[7] Došen, K. Modal logic as metalogic. *SNS-Berichte*, Universität Tübingen, no. 90-5, 1990. To appear in the *Proceedings of the Kleene '90 Conference* (Chaika, Bulgaria).

[8] Došen, K. Sequent systems and groupoid models. *Studia Logica*, **47** (1988), 353–385.

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

[10] Došen, K. & Schroeder-Heister, P. Conservativeness and Uniqueness. *Theoria*, **51** (1985), 159–173.

[11] Došen, K. & Schroeder-Heister, P. (Eds.) *Logics with Restricted Structural Rules: Proceedings of the Conference held in Tübingen, October 1990.* In preparation.

[12] Došen, K. & Schroeder-Heister, P. Uniqueness, definability and interpolation. *Journal of Symbolic Logic*, **53** (1988), 554–570.

[13] Dunn, M. Consecution formulation of positive R with co-tenability and t. In: Anderson, A.R. & Belnap, N.D., *Entailment: The Logic of Relevance and Necessity*, Vol. I, Princeton University Press, 1975, 381–391.

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

[15] Gentzen, G. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, **112** (1935), 493–565.

[16] Giambrone, S. *Gentzen Systems and Decision Procedures for Relevant Logics.* Ph. D. thesis, ANU Canberra, 1983.

[17] Girard, J.-Y. Linear logic. *Theoretical Computer Science*, **50** (1987) 1–102.

[18] Girard, J.-Y., Lafont, Y. & Taylor, P. *Proofs and Types.* Cambridge University Press, 1989.

[19] Hallnäs, L. Generalized Horn Clauses. *SICS Research Report*, no. 86003, 1986.

[20] Hallnäs, L. & Schroeder-Heister, P. A proof-theoretic approach to logic programming. I. Clauses as rules. *Journal of Logic and Computation*, **1** (1990), 261–283; II. Programs as definitions, *ibid.*, in press.

[21] Lambek, J. On the calculus of syntactic types. In: R. Jacobson (ed.), *Proceedings of the 12th Symposium on Applied Mathematics*, AMS, Providence 1961, 166–178, 264–265.

[22] Lambek, J. The mathematics of sentence structure. *American Mathematical Monthly*, **65** (1958), 154–170.

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

[24] Meyer, R. K. Metacompleteness. *Notre Dame Journal of Formal Logic*, **17** (1976), 501–516.

[25] Ono, H. & Komori, Y. Logics without the contraction rule. *Journal of Symbolic Logic*, **50** (1985), 169–201.

[26] Read, S. *Relevant Logic: A Philosophical Examination of Inference.* Oxford: Basil Blackwell, 1988.

[27] Schroeder-Heister, P. An asymmetry between introduction and elimination inferences. In preparation. (Abstract presented at the 9th International Congress of Logic, Methodology and Philosophy of Science, Uppsala, August 1991).

[28] Schroeder-Heister, P. Hypothetical reasoning and definitional reflection in logic programming. In: P. Schroeder-Heister (ed.), *Extensions of Logic Programming. International Workshop, Tübingen, FRG, December 1989, Proceedings.* Springer LNCS, Vol. 475, Berlin (1991), 327–340.

[29] Schroeder-Heister, P. A natural extension of natural deduction. *Journal of Symbolic Logic*, **49** (1984), 1284–1300.

[30] Schroeder-Heister, P. *Structural Frameworks with Higher-Level Rules: Proof-Theoretic Investigations.* Habilitationsschrift. Universität Konstanz, 1987. (Abstract presented at the Workshop on General Logic, Edinburgh, February 1987).

[31] Schroeder-Heister, P. Substructural logics and logic programming. In: L.-H. Eriksson *et al., Proceedings of the Second Workshop on Extensions of Logic Programming, Stockholm 1991*, Springer LNCS, Berlin 1992.

[32] Slaney, J. General logic. *Australian National University, ARP, Technical Report*, No. 4, 1988 (to appear in the *Australasian Journal of Philosophy*).

[33] Wansing, H. The adequacy problem for sequential propositional logic. *ITLI Prepublication Series*, LP-89-07, University of Amsterdam, 1989.