Towards Structurally-Free Theorem Proving

MARCELO FINGER, *Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo, 05508-900, São Paulo, Brazil. Email: mfinger@ime.usp.br*

Abstract

Is it possible to compute in which logics a given formula is deducible? The aim of this paper is to provide a formal basis to answer positively this question in the context of substructural logics. Such a basis is founded on *structurally-free logic*, a logic in which the usual structural rules are replaced by *complex combinator rules*, and thus constitute a generalization of traditional sequent systems.

A family of substructural logics is identified by the set of structural rules admissible to all its members. Combinators encode the sequence of structural rules needed to prove a formula, thus representing the family of logics in which that formula is provable. In this setting, *structurally-free theorem proving* is a decision procedure that inputs a formula and outputs the corresponding combinator when the formula is deducible.

We then present an algorithm to compute a combinator corresponding to a given formula (if it exists) in the fragment containing only the connectives $\rightarrow$ and $\otimes$. The algorithm is based on *equistructural transformations*, *i.e.* it transforms one sequent in a set of simpler sequents from which we can compute the combinator (which represents the structure) of the original sequent. We show that this algorithm is sound and complete and always terminates.

Keywords: Structurally-Free Logic, Automated Deduction, Substructural Logic

1 Introduction

The formulas in a sequent proof can be manipulated by two kinds of rules. *Connective rules* are rules that manipulate sets of premises according to the presence of a connective in some of them. For example, in the sequent calculus, two rules about the $\rightarrow$ connective are:

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi} \ (\vdash \rightarrow) \qquad \frac{\Gamma, \psi \vdash \chi \quad \Delta \vdash \varphi}{\Gamma, \varphi \rightarrow \psi, \Delta \vdash \chi} \ (\rightarrow \vdash)$$

On the other hand, *structural rules* do not refer at all to the existence of connectives in the premises, but operate over the elements of the premises by means of reordering, grouping, duplication, permutation, elimination, etc. In other words, the "structure" of premises is changed through the application of structural rules. For example:

$$\frac{\Gamma, (\varphi, \psi) \vdash \chi}{(\Gamma, \varphi), \psi \vdash \chi} \ (\textit{left-associativity}) \qquad \frac{(\Gamma, \varphi), \psi \vdash \chi}{(\Gamma, \psi), \varphi \vdash \chi} \ (\textit{commutativity})$$

$$\frac{(\Gamma, \varphi), \varphi \vdash \chi}{\Gamma, \varphi \vdash \chi} \ (\textit{contraction}) \qquad \frac{\Gamma \vdash \chi}{\Gamma, \psi \vdash \chi} \ (\textit{weakening or thinning})$$

It is well-known that the several substructural logics differ from each other by virtue of the structural rules allowed in the proofs [10]. Connective rules may be thought of as invariant for all the family of substructural logics; one still has the choice of which connectives are dealt with, *i.e.* which fragment of the logic is on focus. Sometimes, due to the presence of structural rules, some connectives become equivalent and are collapsed.

Historically, however, this was not the way the several substructural logics were created [10]. In fact, Intuitionistic Logic [9] and Relevance Logic [1] were created with clear philosophical motivation, while Linear Logic [14] was proposed for computational reasons. It was the process of understanding these logics that later uncovered similarities and structural differences between them, created several variations, and finally grouped them into the family of substructural logics.

It was quite natural that the original theorem provers for one member of the family concerned only to that member. More recently, theorem provers for substructural logics have been designed to be *modularly extensible* [8]. By modular extensibility, it is meant that the basic mechanism of the theorem prover is common to proofs in several logics, differing only in one "module" of the prover. In this way, for example, [3] provided a modular theorem prover over the $\{\rightarrow, \otimes\}$-fragment of Linear, Relevance and Intuitionistic Logic.

In this work we intend to go one step further and investigate a *structurally-free theorem prover* (SFTP) and provide a formal basis to answer the question:

In which substructural logics is the formula φ a theorem?

Since a set of structural rules identifies the family of substructural logics that admits all such rules, apparently, it would suffice to have a theorem prover that outputs the structural rules used in the proof. But that assumes the prover is based on sequent proofs and accounts for the use of every structural rule, which is not usual. Another way to fake such a prover would be to have at hand a set of theorem provers, each dedicated to one substructural logic. We could then submit φ to each of them and, according to the yes/no answer of each prover, get the set of logics where we can prove φ. To do such a thing would be very inefficient and inelegant. Furthermore, it is always possible to invent new substructural logics and this method would tell us nothing or next to nothing on a new logic.

Ideally, we would like to generate an algorithm, independent from a prover, that given a provable formula in some substructural logic shown by some prover, can indeed "infer" the structural rules that are used to prove such a formula.

Indeed, what we plan to do is to give a mathematical account on how to "compute" a family of logics in which a formula or sequent is provable. For that, we take advantage of a deductive system that has a built in capability of "remembering" the structural rules used. This memory is kept by means of introducing combinators (λ-terms with no free variables) as first class citizens in the logic. Structural rules are replaced by combinator rules in a sequent, thus generalizing the usual sequent system. We then show that some special sequents, called *abstractive sequents*, provide a way to compute what combinator is needed for their deduction. The final part of the method consists on how to *equistructurally* transform a given sequent, *i.e.* a transformation of sequents that preserves their structural/combinatorial content. Those transformations should yield a set of abstractive sequents that allows for the computation of the combinator corresponding to the input sequent.

We say we are still going towards a structurally-free theorem prover because our method does not compute the family of *all logics* in which a formula is provable. However, we guarantee a weaker form of completeness, called *cc-completeness*, that states that one combinator associated to a formula is computed (representing a family of logics not necessarily maximal), if at least one such combinator exists.

The main influence of this work comes from Dunn's and Meyer's work on structurally free logic [11], further developed in [12]. Dunn and Meyer showed how substructural rules are related to λ-calculus combinators, where a combinator is a λ-expression without free variables. Combinators were first introduced by Schönfinkel [19] even before Church created the λ-calculus [5, 6]; it was later shown that combinators could be represented by closed λ-terms (*e.g.* [2] for a proof).

The Curry-Howard isomorphism [17] has already shown a relationship between logic and the λ-calculus, but the relationship investigated by Dunn and Meyer explores a different connection between combinators and substructural logics on the structural side.

Those are certainly not the only connections between combinators and logic. The study of combinators as a foundation of Logic and Mathematics developed into *Combinatorial Logics* [7, 4, 15, 16]. What is significant to our work is that, to provide a foundation for first-order logic, those works of Combinatory Logics introduced other forms of combinators which were not equivalent to λ-terms.

We will also need to introduce other forms of combinators which will not in general be equivalent to λ-terms. The motivation behind their introduction comes from the fact that standard combinators are not sufficient to represent the set of all structural rules of Intuitionistic Logic (see Section 3). The whole set of structure-representation combinators is called *complex combinators*. Given a closed λ-term, there is an algorithm to compute the equivalent combinator, which is known as combinator abstraction [2, 18]. It is also possible to provide a stronger version of combinator abstraction that generates complex combinators [13]. We show that complex combinator abstraction can be directly applied to $\otimes$-sequents (*i.e.* the abstractive sequents mentioned earlier) as a means to compute the associated complex combinator. The whole problem then becomes how to transform a $\{\rightarrow, \otimes\}$-sequents into a set of $\otimes$-sequents.

The paper develops as follows. Section 2 introduces Structurally-Free Logics and gives a Gentzen sequent system for it. Section 3 defines precisely what consists a structurally-free theorem prover and motivates the creation of complex combinators, a theme developed in Section 4. Section 5 develops the notions of equistructurality and shows that abstractive sequents allow for the computation of an associated complex combinator. Section 6 presents an algorithm based on equistructural transformation, used to compute a combinator associated to a formula; soundness, cc-completeness and termination are shown. The paper concludes by pointing out the deficiencies and possible improvements of our method in Section 7.

2 Structurally-Free Logic

Dunn and Meyer [12] proposed a *structurally-free logic* (SFL) where the system is free from any structural pressuposition (whence its name). All structural operations have to be accounted for via *combinators*, *i.e.* λ-terms with no free variables [2]. Figure 1 presents a few examples of combinators. We represent combinators by capital bold letters; the choice of letters is historical.

$\mathbf{B} \equiv \lambda xyz.x(yz)$	$\mathbf{B}xyz \rhd x(yz)$
$\mathbf{C} \equiv \lambda xyz.xzy$	$\mathbf{C}xyz \rhd xzy$
$\mathbf{I} \equiv \lambda x.x$	$\mathbf{I}x \rhd x$
$\mathbf{W} \equiv \lambda xy.xyy$	$\mathbf{W}xy \rhd xyy$
$\mathbf{S} \equiv \lambda xyz.xz(yz)$	$\mathbf{S}xyz \rhd xz(yz)$
$\mathbf{K} \equiv \lambda xy.x$	$\mathbf{K}xy \rhd x$

FIG. 1. λ-calculus combinators

In Figure 1 the symbol $\rhd$ means "reduces to" and in the traditional λ-calculus it is replaced by $=$. In the semantics of SFL [12], inequality is used instead of equality, so we prefer $\rhd$ to $=$. The right hand-side column in Figure 1 shows that combinators can be defined without λ-abstraction and, in this sense, they become *proper* combinators dissociated from the λ-calculus, as in their original formulation [19].

Combinators are not independent from each other. The combinators $\mathbf{W}$ and $\mathbf{S}$ are interdefinable in the presence of the $\mathbf{I}$, $\mathbf{B}$ and $\mathbf{C}$: $\mathbf{W} = \mathbf{CSI}$ and $\mathbf{S} = \mathbf{B}(\mathbf{BW})(\mathbf{BC}(\mathbf{BB}))$ as it can be verified from their definition in the λ-calculus in Figure 1. Indeed, any λ-definable function (and therefore any combinator) can be expressed in terms of the combinators $\mathbf{S}$ and $\mathbf{K}$ [2]; however, the set of combinators presented in Figure 1 is very convenient to account for the use of the most common structural rules and so we name them *primitive combinators*.

Any combinator can be used to represent a structural rule in SFL. Figure 2 shows a Gentzen system for SFL based on a fragment containing only operator $\rightarrow$ (right implication) and $\otimes$ (intensional or multiplicative conjunction, with Girard's Linear Logic notation). Before we discuss Figure 2 in detail, let us present some formal definitions.

The vocabulary of the language consists of a countable set of propositional letters $\mathcal{P} = \{p_1, p_2, \ldots\}$, the set of additional combinator symbols $\mathbf{X}_1, \ldots, \mathbf{X}_i, \ldots$ and the binary connectives $\rightarrow$ and $\otimes$. Any propositional letter and any combinator is a formula; and if φ and ψ are formulas so are $(\varphi \rightarrow \psi)$ and $(\varphi \otimes \psi)$ (parenthesis are omitted, as usual, when no ambiguity is implied).

We then define a sequent in the following way. Any formula is a *structure*; if Γ and Δ are structures, so is the ordered pair (Γ, Δ). We denote structures by capital Greek letters. Structures are therefore binary trees, and the order and nesting of the formulas in them are very important; the ',' in a structure is left-associative, so $\Phi, \Psi, \Xi \equiv (\Phi, \Psi), \Xi$.

A *sequent* $\Gamma \vdash \varphi$ is such that Γ is a structure and φ is a formula; Γ is the sequent's *antecedent* and φ its *succedent*.

In Figure 2 the symbol $\Gamma[\Phi]$ is used to represent one occurrence of Φ inside Γ. Note that, instead of the usual structural rules there is a generic combinator rule that introduce combinators on the antecedent, where $\mathbf{X}$ denotes a generic combinator. To understand its meaning, let us define its symbols precisely. By $\Sigma(\Phi_1, \ldots, \Phi_k)$ it is meant the class of structures that can be constructed from some non-empty subset of $\{\Phi_1, \ldots, \Phi_k\}$; inductively, this is defined as:

- $\Phi_i \in \Sigma(\Phi_1, \ldots, \Phi_k)$ for $1 \leq i \leq k$;

$$\frac{}{\varphi \vdash \varphi}\ \text{(Reflexivity)}$$

Connective Rules

$$\frac{\Gamma \vdash \varphi \quad \Delta \vdash \chi}{\Gamma, \Delta \vdash \varphi{\otimes}\chi}\ (\vdash \otimes) \qquad \frac{\Gamma[\varphi, \psi] \vdash \chi}{\Gamma[\varphi{\otimes}\psi] \vdash \chi}\ (\otimes \vdash)$$

$$\frac{\Gamma, \varphi \vdash \chi}{\Gamma \vdash \varphi{\to}\chi}\ (\vdash \to) \qquad \frac{\Gamma \vdash \varphi \quad \Delta[\psi] \vdash \chi}{\Delta[\varphi{\to}\psi, \Gamma] \vdash \chi}\ (\to \vdash)$$

Generic Combinator Rule

$$\frac{\Gamma[\sigma(\Phi_1, \ldots, \Phi_k)] \vdash \chi}{\Gamma[\mathbf{X}, \Phi_1, \ldots, \Phi_k] \vdash \chi}\ (\mathbf{X} \vdash)$$

FIG. 2. A Gentzen system presentation of SFL

• if Γ and Δ are in $\Sigma(\Phi_1, \ldots, \Phi_k)$, so is (Γ, Δ).

Then $\sigma(\Phi_1, \ldots, \Phi_k)$ denotes an element of $\Sigma(\Phi_1, \ldots, \Phi_k)$. Although the presentation in Figure 2 allows for any combinator rule, some combinators are more useful in the representation of the common structural rules. For instance, Figure 3 presents several examples of combinator rules and their associated "structural" role for the primitive combinators.

The **B** combinator accounts for associativity, **C** for commutativity, **I** for identity, **W** for contraction, **K** for thinning and **S** is also a type of contraction, called factoring.

Given the sequent rules in Figures 2 and 3, we define a *deduction* as a tree such that every leaf node is obtained by the Reflexivity rule, and any internal node is obtained by the application of a sequent rule to its children nodes. A sequent is *deducible* if it is the root of some deduction.

For example, to show that

$$\mathbf{B} \vdash (p{\to}q){\to}[(r{\to}p){\to}(r{\to}q)]$$

we perform the following deduction steps:

$$\cfrac{\cfrac{\cfrac{\cfrac{q \vdash q \quad p \vdash p}{p{\to}q, p \vdash q}\ (\to \vdash) \qquad r \vdash r}{p{\to}q, (r{\to}p, r) \vdash q}\ (\to \vdash)}{\mathbf{B}, p{\to}q, r{\to}p, r \vdash q}\ (\mathbf{B} \vdash)}{\mathbf{B} \vdash (p{\to}q){\to}[(r{\to}p){\to}(r{\to}q)]}\ (\vdash \to\ \ 3\times)$$

This deduction shows that the formula $(p{\to}q){\to}[(r{\to}p){\to}(r{\to}q)]$ is a theorem of the $\to$-fragment of all logics which permit the structural rule for left-associativity. Dunn

identity:	$\dfrac{\Gamma[\Phi] \vdash \chi}{\Gamma[\mathbf{I}, \Phi] \vdash \chi}$	$(\mathbf{I} \vdash)$
left-associativity:	$\dfrac{\Gamma[\Phi, (\Psi, \Xi)] \vdash \chi}{\Gamma[\mathbf{B}, \Phi, \Psi, \Xi] \vdash \chi}$	$(\mathbf{B} \vdash)$
commutativity:	$\dfrac{\Gamma[\Phi, \Xi, \Psi] \vdash \chi}{\Gamma[\mathbf{C}, \Phi, \Psi, \Xi] \vdash \chi}$	$(\mathbf{C} \vdash)$
contraction:	$\dfrac{\Gamma[\Phi, \Psi, \Psi] \vdash \chi}{\Gamma[\mathbf{W}, \Phi, \Psi] \vdash \chi}$	$(\mathbf{W} \vdash)$
factoring:	$\dfrac{\Gamma[\Phi, \Xi, (\Psi, \Xi)] \vdash \chi}{\Gamma[\mathbf{S}, \Phi, \Psi, \Xi] \vdash \chi}$	$(\mathbf{S} \vdash)$
thinning:	$\dfrac{\Gamma[\Phi] \vdash \xi}{\Gamma[\mathbf{K}, \Phi, \Psi] \vdash \xi}$	$(\mathbf{K} \vdash)$

FIG. 3. Examples of combinator rules

and Meyer [12] have shown that the Multiple Cut Rule is admissible in SFL, *i.e.* if added to SFL, no new sequents would be deduced:

$$\frac{\Gamma \vdash \varphi \quad \Delta(\varphi) \vdash \psi}{\Delta(\Gamma) \vdash \psi} \quad (Cut)$$

where the notation $\Delta(\varphi)$ identifies one or more occurrence of φ in Δ (but not necessarily all occurrences) and $\Delta(\Gamma)$ is obtained by replacing those occurrences with Γ. From the admissibility of Multiple Cut follows the admissibility of the single Cut.

Furthermore, what is peculiar to SFL is that combinators are first class citizens, and it is possible to deduce, for example

$$\mathbf{K} \vdash \mathbf{K}$$

with a single application of the reflexivity rule; reflexivity remains the sole pure structural rule preserved in SFL, and sequents thus obtained are called *axioms*. Besides the primitive combinators, we can extend the vocabulary with compound combinators, defined as: (*i*) every primitive combinator is a combinator; (*ii*) if $\mathbf{X}_1$ and $\mathbf{X}_2$ are combinators, so is $\mathbf{X}_1\mathbf{X}_2$. We also abbreviate the structure $(\mathbf{X}_1, \mathbf{X}_2)$ as $\mathbf{X}_1\mathbf{X}_2$ when $\mathbf{X}_1$ and $\mathbf{X}_2$ are combinators (this will simplify the definition of a purely logical sequents below). Given a set of combinators Θ, $\mathbf{X}$ is *based on* Θ if it is an element of Θ, or composed from combinators based on Θ.

Indeed, SFL is not a single logic, but a family of logics. Let SFL(Θ) be the logic SFL which allows only combinators based on Θ; we will abbreviate SFL$^* =$ SFL($\{\mathbf{S}, \mathbf{K}, \mathbf{I}, \mathbf{B}, \mathbf{C}, \mathbf{W}\}$). We can invent logics that only accept certain (primitive or compound) combinators; for example, SFL($\{\mathbf{B}, \mathbf{CB}\}$) accepts commutativity just in the form **CB**. Since $\mathbf{CB}\varphi_1\varphi_2\varphi_3 \rhd \varphi_2(\varphi_1\varphi_3)$, the corresponding combinator rule[1] $(\mathbf{CB}\vdash)$ is:

$$\frac{\Gamma[\Phi_2, (\Phi_1, \Phi_3)] \vdash \chi}{\Gamma[\mathbf{CB}, \Phi_1, \Phi_2, \Phi_3] \vdash \chi}\ (\mathbf{CB}\vdash)$$

In general, we say that for a (primitive or compound) combinator **X** such that $\mathbf{X}x_1 \dots x_k \rhd \sigma(x_1, \dots, x_k)$, its *corresponding combinator rule* and its *corresponding structural rule* are, respectively,

$$\frac{\Gamma[\sigma(\Phi_1, \dots, \Phi_k)] \vdash \chi}{\Gamma[\mathbf{X}, \Phi_1, \dots, \Phi_k] \vdash \chi}\ (\mathbf{X}\vdash) \qquad \text{and} \qquad \frac{\Gamma[\sigma(\Phi_1, \dots, \Phi_k)] \vdash \chi}{\Gamma[\Phi_1, \dots, \Phi_k] \vdash \chi}.$$

As usual, a structural rule is called *admissible* in a logic $\mathcal{L}$ if, when added to logic $\mathcal{L}$, no new sequents are deduced. It is obvious that a combinator rule for a combinator based on Θ is admissible in SFL(Θ). It follows that a combinator rule corresponding to an **X** equivalent to a combinator based on Θ, even if **X** itself is not based on Θ, is admissible in SFL(Θ); for example, $(\mathbf{W}\vdash)$ is admissible in SFL($\{\mathbf{S}, \mathbf{I}, \mathbf{C}\}$) because $\mathbf{W} = \mathbf{CSI}$.

There are certain kinds of SFL formulas and sequents which will play a special role once we start discussing theorem proving. Let a *pure formula/sequent* be one such that it contains no occurrence of a combinator. A pure formula φ is a *structurally-free theorem (sf-theorem)* in SFL(Θ) if there exists a combinator **X** based on Θ such that $\mathbf{X} \vdash \varphi$ is deducible. In this case, **X** is said to be a combinator for φ; Dunn and Meyer [12] call $\mathbf{X} \vdash \varphi$ a *purely logical sequent*[2] (we will refer to a sequent of the form $\mathbf{X}, \varphi_1, \dots, \varphi_n \vdash \chi$ as purely logical as well, for φ_i and χ pure formulas; it is obvious that the latter version can be easily transformed into the former using $(\vdash \rightarrow)$). Purely logical sequents have "pure" deductions, as shown below.

Lemma 2.1 *Every deducible purely logical sequent has a deduction where combinators do not occur in a $\otimes$-formula or $\rightarrow$-formula.*

Proof. Note that all sequent rules in Figure 2 are subformula preserving, that is, if φ_1 is a subformula of φ_2 before the application of the rule, it will remain so after the application of the rule. It follows that if a combinator occurs in a $\otimes$-formula or $\rightarrow$-formula in an intermediary sequent in a deduction, it will remain so in the root of the deduction, and the deduced sequent cannot be purely logical. Thus there must be a deduction of a purely logical sequent without such occurrences. ∎

3 Structurally-Free Theorem Proving

The problem of standard substructural theorem proving is to decide, given a set of premises $\varphi_1, \dots, \varphi_n$ and conclusion χ, if it can be deduced in substructural logic $\mathcal{L}$ that $\varphi_1, \dots, \varphi_n \vdash \chi$. In the light of structurally-free logics, we have a new class of problems.

[1] The combinator **CB** is sometimes called $\mathbf{B}'$

[2] In fact, **X** in $\mathbf{X} \vdash \varphi$ is a structure containing only combinators, but recall that the structure $(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n)$ is abbreviated by $\mathbf{X} = \mathbf{X}_1\mathbf{X}_2 \dots \mathbf{X}_n$; otherwise we would have to define — an carry it on throughout the paper — a purely logical sequent as $\mathbf{X}_1, \dots, \mathbf{X}_n \vdash \varphi$.

Definition 3.1 (SFTP) *A* Structurally-free theorem prover (SFTP) *is an algorithm that takes as input a set of pure formulas, namely a set of premises $\varphi_1, \dots, \varphi_n$ and a conclusion χ and decides whether there exists a combinator $\mathbf{X}$ such that $\mathbf{X}, \varphi_1, \dots, \varphi_n \vdash \chi$ is deducible. The output consists of one such $\mathbf{X}$ in case one exists.*

If we can obtain one such $\mathbf{X}$, we can determine a family of substructural logics in which $\varphi_1, \dots, \varphi_n \vdash \chi$ holds. To support such claim, we prove the following.

Theorem 3.2 (Structure Correspondence) *Any sf-theorem in SFL(Θ) is a theorem in any substructural logic that admits all the structural rules corresponding to the combinators in Θ.*

Proof. Let $\mathcal{L}$ be a substructural logic in which all combinators in Θ are admissible. Let φ be a theorem in SFL(Θ). By Lemma 2.1 there exists a deduction of $\mathbf{X} \vdash \varphi$ without combinators occurring in $\otimes$- and $\rightarrow$-formulas. Every combinator introduction in such deduction can be replaced by an application of the corresponding structural rule admissible in $\mathcal{L}$, thus generating a deduction for $\vdash \varphi$. ∎

This can be applied to several well-known logics.

Corollary 3.3

(a) every SFL* sf-theorem is a theorem of Intuitionistic Logic.

(b) every SFL($\{\mathbf{I}, \mathbf{B}, \mathbf{C}\}$) sf-theorem is a theorem of Linear Logic.

(c) every SFL($\{\mathbf{I}, \mathbf{B}, \mathbf{C}, \mathbf{W}\}$) sf-theorem is a theorem of Relevance Logic.

(d) every SFL($\{\mathbf{I}\}$) sf-theorem is a theorem of the non-associative Lambek Calculus.

Indeed, assume $\mathbf{X} \vdash \varphi$ is a deducible purely logical sequent. Linear Logic allows only structural rules of identity, associativity and commutativity, but these rules can be used any number of times and in any order; so if $\mathbf{X}$ contains only $\mathbf{I}$'s, $\mathbf{B}$'s and $\mathbf{C}$'s then we know that $\vdash \varphi$ holds in Linear Logic. Relevance Logic additionally accepts structural rules for contraction ($\mathbf{W}$ or $\mathbf{S}$), and Intuitionistic Logic accepts contraction ($\mathbf{W}$ or $\mathbf{S}$) and weakening ($\mathbf{K}$). The non-associative Lambek calculus allows only $\mathbf{I}$.

Before we proceed explaining *how* to construct an SFTP, let us pay some attention to Corollary 3.3, item (a). Indeed, let us consider the validity of its reciprocal, namely is it the case that a theorem of the $(\rightarrow, \otimes)$-fragment of Intuitionistic Logic (IL) is also an sf-theorem in SFL*? The answer is no, and to see why it suffices to note that $\varphi_1 \otimes (\varphi_2 \otimes \varphi_3) \rightarrow (\varphi_1 \otimes \varphi_2) \otimes \varphi_3$ is an IL theorem, but there is no combinator $\mathbf{X}$ such that $\mathbf{X} \vdash \varphi_1 \otimes (\varphi_2 \otimes \varphi_3) \rightarrow (\varphi_1 \otimes \varphi_2) \otimes \varphi_3$. There is no combinator definable in terms of a closed λ-term corresponding to the right-associativity structural rule:

$$\frac{\Gamma[\varphi_1, \varphi_2, \varphi_3] \vdash \chi}{\Gamma[\varphi_1, (\varphi_2, \varphi_3)] \vdash \chi} \quad (\textit{right-associativity})$$

(for a proof of this fact, see [13].) To eliminate such deficiency, we introduce *complex combinators*.

4 Complex Combinators

We note that right-associativity is "the inverse" of left-associativity in terms of "functional inverses". So we start by considering inverse combinators[3], defined in terms of reducibility. We denote the functional inverse of combinator $\mathbf{X}$ as $\mathbf{X}^\bullet$.

Definition 4.1 (Inverse Combinators) *Let* $\Theta_0 = \{\mathbf{S}, \mathbf{I}, \mathbf{B}, \mathbf{C}, \mathbf{W}\}$. *Let* $\mathbf{X}$ *be a combinator based on* Θ_0 *such that*

$$\mathbf{X}x_1 \dots x_k \rhd P(x_1, \dots, x_k)$$

where $x_1, \dots, x_k$ *are variables ranging over terms,* $P(x_1, \dots, x_k)$ *is a* λ*-free term built up using all* x_i, $1 \leq i \leq k$. *The* inverse combinator *of* $\mathbf{X}$, *denoted* $\mathbf{X}^\bullet$, *is such that*

$$\mathbf{X}^\bullet * P(x_1, \dots, x_k) \rhd x_1 \dots x_k$$

where $*$ *is the prepending operation, i.e.* $x{*}y = xy$ *for atomic* y, *and* $x{*}(zw) = (x{*}z)w$.

First note that inverse combinators are not always combinators. Consider, for example, the combinator $\mathbf{B}$, $\mathbf{B}xyz \rhd x(yz)$; its inverse is $\mathbf{B}^\bullet$, $\mathbf{B}^\bullet x(yz) \rhd xyz$, is not equivalent to any standard combinator; similarly for $\mathbf{S}^\bullet$ and $\mathbf{W}^\bullet$. However, $\mathbf{C}^\bullet$ is such that $\mathbf{C}^\bullet xyz \rhd xzy$ and indeed $\mathbf{C}^\bullet \equiv \mathbf{C}$; similarly for $\mathbf{I}^\bullet \equiv \mathbf{I}$.

The inverse of $\mathbf{K}$ was ruled out. If $\mathbf{K}^\bullet$ were allowed, it would be possible to build a combinator that reduces to any term. In fact, since for any x and y $\mathbf{K}xy \rhd x$, then $\mathbf{K}^\bullet x \rhd xy$, so $\mathbf{K}^\bullet\mathbf{I} \rhd z$ for any z. This means that $\mathbf{K}^\bullet\mathbf{I}$ is no longer a function and it trivializes reducibility. It may be interesting to study some controlled uses of $\mathbf{K}^\bullet$ (such as $\mathbf{K}^\bullet_y$, $\mathbf{K}^\bullet_y x \rhd xy$), but we will not do it here, opting to rule it out of our language.

Definition 4.1 defines duality for any standard combinator, not only the primitive ones. So, for example, the inverse of $\mathbf{CB}$, where $\mathbf{CB}xyz \rhd y(xz)$, is $(\mathbf{CB})^\bullet$, defined as $(\mathbf{CB})^\bullet y(xz) \rhd xyz$. In fact, $(\mathbf{CB})^\bullet = \mathbf{BB}^\bullet(\mathbf{CI})$ because

$$\mathbf{BB}^\bullet(\mathbf{CI})y(xz) \rhd \mathbf{B}^\bullet(\mathbf{CI}y)(xz) \rhd \mathbf{CI}yxz \rhd \mathbf{I}xyz \rhd xyz$$

$\mathbf{BB}^\bullet(\mathbf{CI})$ is what we term a *complex combinator*.

Definition 4.2 (Complex Combinators) *A* complex combinator *is a term built up in the following way:*

- *every primitive combinator is a complex combinator;*
- *the inverses of the primitive combinators in* Θ_0, *are complex combinator, i.e.* $\mathbf{S}^\bullet$, $\mathbf{I}^\bullet$, $\mathbf{B}^\bullet$, $\mathbf{C}^\bullet$ *and* $\mathbf{W}^\bullet$*;*
- *if* $\mathbf{X}_1$ *and* $\mathbf{X}_2$ *are complex combinators, so is* $(\mathbf{X}_1\mathbf{X}_2)$.

Composition of complex combinators is left-associative, as usual. An inverse-free combinator *is one without the occurrence of an inverse primitive combinators, i.e. a combinator equivalent to a* λ*-term and what we have considered as a combinator before this section.*

[3]We thank a reviewer for pointing out that the term *inverse combinator* has already been used by Curry with a different sense; however, we maintain the current usage for we are really defining functional inverses

One important property of complex combinators is the substitutivity of reduced forms:

Lemma 4.3 *Let* $\mathbf{X}_1$ *and* $\mathbf{X}_2$ *be complex combinators such that* $\mathbf{X}_1 \rhd \mathbf{X}_2$. *Let* P_1 *and* P_2 *be any terms. Then*

$$\mathbf{X}_1 * P_1 \rhd P_2 \quad \textit{iff} \quad \mathbf{X}_2 * P_1 \rhd P_2$$

Proof. By induction on the length of reduction of $\mathbf{X}_1 \rhd \mathbf{X}_2$; details omitted. ∎

We can also have inverses of complex combinators.

Definition 4.4 (Complex Inverses) *Let* $\mathbf{X}$ *and* $\mathbf{X}^\bullet$ *be* $\mathbf{K}$*-free complex combinators;* $\mathbf{X}$ *and* $\mathbf{X}^\bullet$ *are* inverse complex combinators *whenever*

$$\mathbf{X} * P_1(x_1,\ldots,x_k) \rhd P_2(x_1,\ldots,x_k) \quad \textit{iff} \quad \mathbf{X}^\bullet * P_2(x_1,\ldots,x_k) \rhd P_1(x_1,\ldots,x_k)$$

where $x_1,\ldots,x_k$ *are variables ranging over terms,* $P_{1,2}(x_1,\ldots,x_k)$ *are* λ*-free terms built up using all* x_i, $1 \leq i \leq k$.

It follows directly from Definition 4.4 that $(\mathbf{X}^\bullet)^\bullet \equiv \mathbf{X}$. Functions are not invertible in general, but we can take the inverse of any combinator. To solve this apparent paradox, we bring the attention to a peculiar behaviour of inverse primitive combinators. Unlike inverse-free combinators, inverses do not always reduce when applied to a large enough number of terms. For example, $\mathbf{B}^\bullet xyz$ does not reduce. This is due to the fact that inverse primitive combinators perform a kind of *pattern matching*, so $\mathbf{B}^\bullet$ will only reduce when applied to a term that matches the pattern $x(yz)$.

Looking back at the example above that shows $(\mathbf{CB})^\bullet = \mathbf{BB}^\bullet(\mathbf{CI})$, one is left with the question on whether the inverse of any complex combinator is equivalent to a complex combinator. This is not always the case; for instance, combinators without a normal form do not have an inverse. Indeed, $\mathbf{WWW}$ reduces to itself, it does not have a normal form and we cannot apply the definition above. See [13] for a discussion on how to compute inverses for a particularly useful class of complex combinators called *list combinators*. For the moment let us concentrate on adding inverse combinator rules to SFL.

Notation: If σ is an antecedent (or a formula), then $\tilde{\sigma}$ is a combinator term obtained from deleting the symbols ‘,’ and ‘$\otimes$’ from σ.

The general form of a complex combinator $\mathbf{X}$ rule, $\mathbf{X}*\tilde{\sigma}_1(x_1,\ldots,x_k)\rhd\tilde{\sigma}_2(x_1,\ldots,x_k)$, is

$$\frac{\Gamma[\sigma_2(\varphi_1,\ldots,\varphi_k)] \vdash \chi}{\Gamma[\mathbf{X} * \sigma_1(\varphi_1,\ldots,\varphi_k)] \vdash \chi}\ (\mathbf{X}\vdash)$$

For example, the rules for the inverse primitive operators are:

$$\frac{\Gamma[\varphi_1,\varphi_2,\varphi_3] \vdash \chi}{\Gamma[\mathbf{B}^\bullet,\varphi_1,(\varphi_2,\varphi_3)] \vdash \chi}\ (\mathbf{B}^\bullet\vdash) \qquad \frac{\Gamma[\varphi_1,\varphi_2,\varphi_3] \vdash \chi}{\Gamma[\mathbf{S}^\bullet,\varphi_1,\varphi_3,(\varphi_2,\varphi_3)] \vdash \chi}\ (\mathbf{S}^\bullet\vdash)$$

$$\frac{\Gamma[\varphi_1,\varphi_2] \vdash \chi}{\Gamma[\mathbf{W}^\bullet,\varphi_1,\varphi_2,\varphi_2] \vdash \chi}\ (\mathbf{W}^\bullet\vdash)$$

The rules for $\mathbf{C}^\bullet$ and $\mathbf{I}^\bullet$ are omitted because they are identical to those for $\mathbf{C}$ and $\mathbf{I}$. $(\mathbf{B}^\bullet \vdash)$ accounts for right-associativity, $(\mathbf{W}^\bullet \vdash)$ accounts for expansion (the opposite of contraction) and $(\mathbf{S}^\bullet \vdash)$ accounts for distribution (a notion similar to expansion and the converse of factoring). From $\mathbf{K}xyy \rhd xy$ we see that $(\mathbf{W}^\bullet \vdash)$ is admissible in any logic that admits $(\mathbf{K} \vdash)$, but not the other way round. Similarly, $(\mathbf{S}^\bullet \vdash)$ is admissible if $\mathbf{B}$, $\mathbf{B}^\bullet$, $\mathbf{C}$ and $\mathbf{W}^\bullet$ are.

With respect to the proofs of purely logical sequents that include a complex combinator, we note that the proof Lemma 2.1 does not mention any combinator rule in particular, and therefore Lemma 2.1 holds for complex combinators as well. As a consequence, the Structure correspondence Theorem 3.2 also holds after we extend the logic with complex combinator.

We can then go back to the converse of Corollary 3.3(a), which motivated the introduction of complex combinators, and finally prove it.

Theorem 4.5 (IL Correspondence) *A formula φ is deducible in the $(\rightarrow, \otimes)$-fragment of Intuitionistic Logic iff it is an sf-theorem in SFL($\{\mathbf{K}, \mathbf{S}, \mathbf{I}, \mathbf{B}, \mathbf{C}, \mathbf{W}, \mathbf{B}^\bullet, \mathbf{W}^\bullet, \mathbf{S}^\bullet\}$).*

Proof. $(\Leftarrow)$ This is just Corollary 3.3(a), which still holds for complex combinators as we have argued above.

$(\Rightarrow)$ Let $\varphi_1, \ldots, \varphi_n \vdash \chi$ be a deducible sequent in the $(\rightarrow, \otimes)$-fragment of Intuitionistic Logic. We prove by induction on the *length l* of the intuitionistic deduction, *i.e.* the maximum of the number of sequents in each branch of the deduction, that either $\varphi_1, \ldots, \varphi_n \vdash \chi$ is deducible or there exists an $\mathbf{X}$ such that $\mathbf{X}, \varphi_1, \ldots, \varphi_n \vdash \chi$ in SFL. For $l = 1$, the only possible deducible sequent is of the form $\chi \vdash \chi$, also deducible in SFL. So $\mathbf{I}, \chi \vdash \chi$ is an SFL-deducible sequent in the desired format.

The inductive step is shown by cases. If the root sequent is obtained with a connective rule with a single premise, *i.e.* rules $(\otimes \vdash)$ and $(\vdash \rightarrow)$, the same rules are applied in SFL and the induction hypothesis clearly lead us to a root in the desired format; details omitted.

Combinators $\mathbf{K}$, $\mathbf{S}$, $\mathbf{I}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{W}$, $\mathbf{B}^\bullet$, $\mathbf{W}^\bullet$ and $\mathbf{S}^\bullet$ all correspond to structural rules admissible in IL, so if the root sequent is obtained with a structural rule then in SFL the root will be obtained by a combinator rule which always add a combinator to the leftmost position. The induction hypothesis clearly leads us to a root in the desired format.

The interesting cases are the two-premised connective rules $(\vdash \otimes)$ and $(\rightarrow \vdash)$. Suppose the root of the intuitionistic deduction tree is obtained with $(\vdash \otimes)$ from premises $\psi_1, \ldots, \psi_m \vdash \psi$ and $\chi_1, \ldots, \chi_k \vdash \chi$. By the induction hypothesis, there are $\mathbf{X}_1$ and $\mathbf{X}_2$ such that its corresponding SFL deduction will be

$$\frac{\mathbf{X}_1, \psi_1, \ldots, \psi_m \vdash \psi \quad \mathbf{X}_2, \chi_1, \ldots, \chi_k \vdash \chi}{\mathbf{X}_1, \psi_1, \ldots, \psi_m, (\mathbf{X}_2, \chi_1, \ldots, \chi_k) \vdash \psi \otimes \chi} \; (\vdash \otimes)$$

This sequent is not in the desired format because $\mathbf{X}_2$ occurs inside the sequent. But then, by simple combinator manipulation, we know that there always exists a combinator $\mathbf{X}_{\mathbf{B},\mathbf{B}^\bullet,\mathbf{C}}$ built up using only the combinators $\mathbf{B}$, $\mathbf{B}^\bullet$ and $\mathbf{C}$, such that it "moves" $\mathbf{X}_2$ to the front of the sequent by means of simple order permutations:

$$\frac{\mathbf{X}_1, \psi_1, \ldots, \psi_m, (\mathbf{X}_2, \chi_1, \ldots, \chi_k) \vdash \psi \otimes \chi}{\mathbf{X}_{\mathbf{B},\mathbf{B}^\bullet,\mathbf{C}}\mathbf{X}_1\mathbf{X}_2, \psi_1, \ldots, \psi_m, (\chi_1, \ldots, \chi_k) \vdash \psi \otimes \chi} \; (\mathbf{X}_{\mathbf{B},\mathbf{B}^\bullet,\mathbf{C}} \vdash)$$

That such a permutation exists can be shown by a simple induction on $m+k$; details omitted. The final sequent is then in the desired format. Note that by omitting the combinators in this new root, we get the corresponding root in the intuitionistic proof.

Suppose the root of the intuitionistic deduction tree is obtained with $(\rightarrow\vdash)$ from premises $\psi_1,\ldots,\psi_m\vdash\psi$ and $\Gamma[\varphi]\vdash\chi$. By the induction hypothesis, its corresponding SFL deduction will be

$$\frac{\mathbf{X}_1,\psi_1,\ldots,\psi_m\vdash\psi \quad \Gamma[\varphi]\vdash\chi}{\Gamma[\psi\rightarrow\varphi,(\mathbf{X}_1,\psi_1,\ldots,\psi_m)]\vdash\chi}\ (\rightarrow\vdash)$$

for some $\mathbf{X}_1$, clearly not in the desired format for $\mathbf{X}_1$ occurs in the middle of the sequent. But again, by a similar argument, there exists a combinator $\mathbf{X}_{\mathbf{B},\mathbf{B}^\bullet,\mathbf{C}}$ which permutes $\mathbf{X}_1$ such that

$$\mathbf{X}_{\mathbf{B},\mathbf{B}^\bullet,\mathbf{C}}\mathbf{X}_1\Gamma[\psi\rightarrow\varphi,(\psi_1,\ldots,\psi_m)]\vdash\chi$$

in the desired format; by deleting the combinators from it we get the root in the intuitionistic proof. ∎

It is easy to generalize the above result for other logics.

Theorem 4.6 (Full Correspondence) *Let Θ be a set of complex combinators containing $\mathbf{I}$, $\mathbf{B}$, $\mathbf{B}^\bullet$ and $\mathbf{C}$. Let $\mathcal{L}(\Theta)$ be the class of substructural logics that accepts the structural rules corresponding to the combinators in Θ. The following are equivalent:*

(a) φ is an sf-theorem in SFL(Θ).

(b) φ is a theorem in the $\{\rightarrow,\otimes\}$-fragment of all logics in $\mathcal{L}(\Theta)$.

The proof is totally analogous to that of Theorem 4.5. Bearing in mind Theorem 4.6 and that Cut Elimination holds for Intuitionistic Logic, the following property comes with little surprise.

Lemma 4.7 *Multiple Cut remains an admissible rule in SFL(Θ).*

To prove the result above one would have to modify the analogous proof given in [12], just adding three more cases of cut elimination for the three new combinator rules. Such a task is straightforward but tedious; we omit the details.

We concentrate now on the construction of an SFTP. For the rest of this paper, unless otherwise specified, whenever we mention combinators we actually mean *complex* combinators. By SFL we mean SFL($\{\mathbf{K},\mathbf{S},\mathbf{I},\mathbf{B},\mathbf{C},\mathbf{W},\mathbf{B}^\bullet,\mathbf{W}^\bullet,\mathbf{S}^\bullet\}$).

5 Equistructurality

The problem we focus on is how to compute the combinator for an sf-theorem without generating a sequent proof for it. For that, we need first a few definitions.

Definition 5.1 (Equistructurality) *Two sf-theorems φ and ψ are* equistructural *if there exists a combinator $\mathbf{X}$ such that*

$$\mathbf{X}\vdash\varphi \text{ and } \mathbf{X}\vdash\psi$$

are deducible in SFL. Similarly, two pure sequents $\varphi_1, \ldots, \varphi_m \vdash \varphi$ *and* $\psi_1, \ldots, \psi_n \vdash \psi$ *are* equistructural *if*

$$\mathbf{X}, \varphi_1, \ldots, \varphi_m \vdash \varphi \text{ and } \mathbf{X}, \psi_1, \ldots, \psi_n \vdash \psi$$

are deducible in SFL. Also, a pure sequent $\Gamma \vdash \varphi$ *is* equistructural *to a set of sequents* $S = \{\Gamma_i \vdash \psi_i | 1 \leq i \leq k\}$ *if there are combinators* $\mathbf{X}$ *and* $\mathbf{X}_1, \ldots, \mathbf{X}_k$ *such that* $\mathbf{X}_i * \Gamma_i \vdash \psi_i$ *are deducible in SFL and*

$$\frac{\mathbf{X}_1 * \Gamma_1 \vdash \psi_1 \quad \ldots \quad \mathbf{X}_k * \Gamma_k \vdash \psi_k}{\mathbf{X} * \Gamma \vdash \varphi}$$

is deducible in SFL; $\mathbf{X}$ *is thus computable from* $\mathbf{X}_1, \ldots, \mathbf{X}_k$*.*

Equistructurality plays an important role in our method to create an SFTP. Let $\mathbf{X}$ be an unknown combinator (*i.e.* a variable ranging over the set of admissible combinators) that we want to compute to make $\mathbf{X}, \varphi_1, \ldots, \varphi_m \vdash \chi$ deducible. What we propose to do is to manipulate the φ_i and χ to obtain a "more adequate" equistructural sequent. But what is an "adequate" sequent? We show that a sequent is adequate if it is in the $\otimes$-fragment of the language. Later manipulations will transform a sequent in the $\{\otimes, \rightarrow\}$-language into a set of sequents in the $\otimes$-fragment.

Properties of the $\otimes$-fragment

The main attraction of the $\otimes$-fragment is that in it we can compute the combinator associated with a sequent, as shown below.

Lemma 5.2 *Let* $p_1, \ldots, p_k$ *be propositional letters and let* $\chi(p_1, \ldots, p_k)$ *be a formula built up using* $\otimes$ *and* p_i*, possibly with omissions. Then*

$$\mathbf{X}, p_1, \ldots, p_k \vdash \chi(p_1, \ldots, p_k)$$

is deducible for $\mathbf{X} = \lambda p_1 \ldots p_k . \tilde{\chi}(p_1, \ldots, p_k)$*, where* $\tilde{\chi}(p_1, \ldots, p_k)$ *is obtained from* $\chi(p_1, \ldots, p_k)$ *by deleting the* $\otimes$*'s.*

Proof. We know that for every λ-term without free variables, there exists a combinator, built up using primitive combinators **S**, **K**, **I**, **B**, **C** and **W**, that is equivalent to it [2]. In particular, let $\mathbf{X}$ be such that $\mathbf{X}p_1 \ldots p_k \rhd \tilde{\chi}(p_1, \ldots, p_k)$; $\mathbf{X}$ can be seen as a term built up from primitive combinators $\mathbf{X}_1, \ldots \mathbf{X}_l$. We construct bottom-up the deduction of $\mathbf{X}, p_1, \ldots, p_k \vdash \chi(p_1, \ldots, p_k)$. At each step, the antecedent of the lower sequent is changed by applying the innermost-leftmost primitive combinator $\mathbf{X}_i$ in $\mathbf{X}$, generating the upper sequent; it is clear this corresponds to the use of the $(\mathbf{X}_i \vdash)$ rule. For example,

$$\mathbf{BKC}p_1p_2p_3p_4 \rhd \mathbf{K}(\mathbf{C}p_1)p_2p_3p_4 \rhd \mathbf{C}p_1p_3p_4 \rhd p_1p_4p_3$$

generates, bottom-up, the deduction tree

$$\cfrac{\cfrac{\cfrac{p_1, p_4, p_3 \vdash p_1 \otimes p_4 \otimes p_3}{\mathbf{C}, p_1, p_3, p_4 \vdash p_1 \otimes p_4 \otimes p_3}\;(\mathbf{C}\vdash)}{\mathbf{K}, (\mathbf{C}, p_1), p_2, p_3, p_4 \vdash p_1 \otimes p_4 \otimes p_3}\;(\mathbf{K}\vdash)}{\mathbf{BKC}, p_1, p_2, p_3, p_4 \vdash p_1 \otimes p_4 \otimes p_3}\;(\mathbf{B}\vdash)$$

The succedent never changes. At the top of the deduction tree we find a sequent of the form $\sigma_\chi(p_1,\ldots,p_k) \vdash \chi(p_1,\ldots,p_k)$, where $\sigma_\chi(p_1,\ldots,p_k)$ is obtained from $\chi(p_1,\ldots,p_k)$ by replacing $\otimes$'s with separating commas; this is clearly a deducible sequent and the deduction tree is finished. ■

A sequent of the format of Lemma 5.2 is called a *λ-abstractive sequent*. The combinator $\mathbf{X}$ obtained with a λ-abstractive sequent is inverse-free, *i.e.* it is a non-complex combinator. We can extend the lemma to cope with complex combinators as well.

Lemma 5.3 *Let $P = \{p_1,\ldots,p_k\}$ and $Q = \{q_1,\ldots,q_l\}$ be sets of propositional letters such that $Q \subseteq P$. Let $\sigma(p_1,\ldots,p_k)$ be a structure containing all p_i's at least once, and let $\chi(q_1,\ldots,q_l)$ be a formula built using $\otimes$ and the elements of Q. Then*

$$\mathbf{X} * \sigma(p_1,\ldots,p_k) \vdash \chi(q_1,\ldots,q_l)$$

is deducible in SFL for $\mathbf{X} * \tilde{\sigma}(p_1,\ldots,p_k) \rhd \tilde{\chi}(q_1,\ldots,q_l)$,

Proof. Since $Q \subseteq P$, it is guaranteed that there exists a complex combinator $\mathbf{X}$ such that $\mathbf{X} * \tilde{\sigma}(p_1,\ldots,p_k) \rhd \tilde{\chi}(q_1,\ldots,q_l)$; $\mathbf{X}$ can be effectively computed [13]. Then, by an analogous construction to that of Lemma 5.2 we obtain a deduction for $\mathbf{X} * \sigma(p_1,\ldots,p_k) \vdash \chi(q_1,\ldots,q_l)$. ■

A sequent of the format of Lemma 5.3 is called a *(complex combinator) abstractive sequent*; λ-abstractive sequents are always abstractive. We can then summarize the computational properties of the $\otimes$-fragment.

Theorem 5.4 *In the $\otimes$-fragment, $\mathbf{X} * \Gamma \vdash \chi$ is deducible in SFL iff $\mathbf{X} * \tilde{\Gamma} \rhd \tilde{\chi}$.*

Proof. Suppose $\mathbf{X} * \Gamma \vdash \chi$. It suffices to note that inference rules of (Reflexivity), $(\vdash \otimes)$, $(\otimes \vdash)$ and the combinator rules are all preserved under $\rhd$. Then, by a simple induction on the derivation of $\mathbf{X} * \Gamma \vdash \chi$ we get $\mathbf{X} * \tilde{\Gamma} \rhd \tilde{\chi}$.

Suppose now $\mathbf{X} * \tilde{\Gamma} \rhd \tilde{\chi}$. Then by the argument used in the proofs of Lemmas 5.2 and 5.3, a proof for $\mathbf{X} * \Gamma \vdash \chi$ is constructed. ■

The following corollary implies that we can prefer to compute a more simplified (irreducible) form of combinator without loss of expressivity.

Corollary 5.5 *Let $\mathbf{X}_1$ and $\mathbf{X}_2$ be complex combinators.*

(a) Suppose $\mathbf{X}_1 \rhd \mathbf{X}_2$. Then, $\mathbf{X}_1 * \Gamma \vdash \chi$ is deducible in the $\otimes$-fragment iff $\mathbf{X}_2 * \Gamma \vdash \chi$ is.

(b) Suppose that $\mathbf{X}_1, \Phi_1,\ldots,\Phi_n, q \vdash \chi \otimes q$ and $\mathbf{X}_2, \Phi_1,\ldots,\Phi_n \vdash \chi$ are abstractive sequents where q occurs neither in $(\Phi_1,\ldots,\Phi_n)$ nor in χ. Then $\mathbf{X}_2, \Phi_1,\ldots,\Phi_n, q \vdash \chi \otimes q$.

Proof.

(a)
$$\begin{array}{llll} \mathbf{X}_1 * \Gamma \vdash \chi \text{ is deducible} & \text{iff} & \mathbf{X}_1 * \tilde{\Gamma} \rhd \tilde{\chi} & \text{by Theorem 5.4} \\ & \text{iff} & \mathbf{X}_2 * \tilde{\Gamma} \rhd \tilde{\chi} & \text{by Lemma 4.3} \\ & \text{iff} & \mathbf{X}_2 * \Gamma \vdash \chi \text{ is deducible} & \text{by Theorem 5.4} \end{array}$$

(b) ¿From $\mathbf{X}_2, \Phi_1,\ldots,\Phi_n \vdash \chi$ and Theorem 5.4 we get $\mathbf{X}_2 \tilde{\Phi}_1,\ldots,\tilde{\Phi}_n \rhd \tilde{\chi}$. Then clearly $\mathbf{X}_2 \tilde{\Phi}_1,\ldots,\tilde{\Phi}_n, q \rhd \tilde{\chi} q$ and again by Theorem 5.4 we get $\mathbf{X}_2, \Phi_1,\ldots,\Phi_n, q \vdash \chi \otimes q$. ■

We can use Corollary 5.5(b) to compute a simpler combinator ($\mathbf{X}_2$) instead of a more complex one ($\mathbf{X}_1$). We will make use of this simplification in the examples below.

If it is possible to obtain a λ-abstractive sequent equistructural to a given sequent, there are known algorithms that perform *combinator abstraction* [18], *i.e.* given a λ-term without free variables, it outputs an equivalent combinator in terms of the primitive combinators. If only **S** and **K** are allowed as primitive, the size of the output combinator (*i.e.* the number of primitive combinators in it) is exponential with k, the number of bound λ-variables. If **I**, **B** and **C** are also allowed, the output size decreases and becomes quadratic with k; see [18].

Similarly, it is possible to perform complex-combinator abstraction. Given an abstractive sequent, by interpreting complex combinators into their corresponding structural rules, complex-combinator abstraction gives us the logics in which it is possible to prove this sequent (and all other sequents equistructural to it). Details on complex-combinator abstraction can be found in [13].

We now turn to the problem of computing a corresponding complex combinator for a given pure sequent, the core of SFTP.

6 Equistructural Transformation Rules

A *deducible pure sequent* $\Gamma \vdash \chi$ is a pure sequent such that there exists a combinator $\mathbf{X}$, $\mathbf{X} * \Gamma \vdash \chi$ is deducible in SFL. $\mathbf{X}$ is called the sequent's *corresponding combinator*; a deducible pure sequent can, in principle, have more than one corresponding combinator. At this stage, we will be satisfied if we can compute one such combinator, if it exists. In other words, we want a decision procedure that outputs a complex combinator associated with a pure sequent if it is deducible.

The basic idea is, given $\mathbf{X} * \Gamma \vdash \chi$ with $\mathbf{X}$ unknown, to *equistructurally transform* this sequent into a set of sequents $S = \{\mathbf{X}_i * \Gamma_i \vdash \chi_i | 1 \leq i \leq k\}$ and $\mathbf{X} = f(\mathbf{X}_1, \ldots, \mathbf{X}_k)$, such that we know how to compute $\mathbf{X}$ from the combinators in S. We apply such transformations to obtain $\rightarrow$-free sequents. We already know how to decide if an $\rightarrow$-free sequent is deducible (just check if it is abstractive) and how to compute its corresponding combinator using complex combinator abstraction (Theorem 5.4). This idea is expressed in the basic SFTP algorithm:

Algorithm 6.1 Input: *pure sequent* $\Gamma \vdash \chi$
Output: *a corresponding combinator* $\mathbf{X}$*, if one exists; "fail" otherwise.*

```
S := {$\mathbf{X} * \Gamma \vdash \chi$}
V := {$\mathbf{X}$}
While S is not empty
   Choose a sequent $\mathbf{X}_i * \Gamma_i \vdash \chi_i$ from S
   if $\mathbf{X}_i * \Gamma_i \vdash \chi_i$ is $\rightarrow$-free
      if it is abstractive
         Compute $\mathbf{X}_i$ using combinator abstraction
         V := V[$\mathbf{X}_i$/value($\mathbf{X}_i$)] - {$\mathbf{X}_i$} ($\mathbf{X}_i$ is removed from V
            and its computed value substituted)
      else return(''fail'') (input sequent is not derivable)
   else
```

```
        transform $\mathbf{X}_i * \Gamma_i \vdash \chi_i$ into set $S' = \{\mathbf{Y}_j * \Gamma_j \vdash \chi_j | 1 \leq j \leq k_i\}$
        S := S ∪ S' - $\{\mathbf{X}_i * \Gamma_i \vdash \chi_i\}$
        V := V ∪ $\{\mathbf{X}_i = f(\mathbf{Y}_1, \ldots, \mathbf{Y}_{k_i})\}$ ∪ $\{\mathbf{Y}_j | 1 \leq j \leq k_i\}$
return($\mathbf{X}$)
end.
```

Note that all combinators $\mathbf{X}_i$ in S and V are unknown variables, and as soon as a value for them is computed, their value is substituted in those sets and their value is removed. At the end, we should have just the initial combinator value $\mathbf{X}$, with its computed value.

Algorithm 6.1 needs to be complemented by a set of transformation rules to transform $\mathbf{X}_i * \Gamma_i \vdash \chi_i$ into a set of sequents, each of which contains fewer number of $\rightarrow$-connectives. We would like rules that force Algorithm 6.1 to have the basic properties of *termination*, *correctness* and *completeness*. Let us first define precisely what we mean by those properties.

By *termination* we mean that the algorithm neither gets into an infinite loop nor gets stuck without a rule to apply when there is still a sequent in S.

The algorithm is correct if it outputs only correct combinators:

Definition 6.2 *Given an input pure sequent $\Gamma \vdash \chi$ such that the transformations computed a combinator $\mathbf{X}$, this computation is* correct *if the sequent $\mathbf{X} * \Gamma \vdash \chi$ is derivable in SFL.*

Lemma 6.3 *A sufficient condition for correctness of Algorithm 6.1 is that all transformation rules generate a set of sequents that is equistructural to the sequent being transformed.*

Proof. By induction on the deduction of $\mathbf{X} * \Gamma \vdash \chi$, directly from the definition of equistructurality in Definition 5.1. If a combinator can be computed, it means that $\mathbf{X} * \Gamma \vdash \chi$ can be derived in SFL from axioms and abstractive sequents. ■

As for completeness, at this point we do not aim at developing an algorithm that is strongly complete. Strong completeness would mean that we could compute all the combinators corresponding to a given sequent. More modestly, we investigate weaker forms of completeness.

Definition 6.4 *Consider an input pure sequent $\Gamma \vdash \chi$ such that, there is an SFL derivation for $\mathbf{X} * \Gamma \vdash \chi$ for some $\mathbf{X}$. Algorithm 6.1 is* cc-complete *if it can compute one such $\mathbf{X}$ using the given set of transformations.*

It follows from Theorem 4.5 that cc-completeness implies that if a sequent is derivable in the $\{\otimes, \rightarrow\}$-fragment of Intuitionistic Logic, then we can compute an associated complex combinator. Also, it means that if the algorithm fails, there is no derivation of the input sequent for any combinator.

Lemma 6.5 *Suppose we have a set of transformation rules that are correct and terminate. A sufficient condition for these rules to generate a cc-complete algorithm is that whenever a transformation rule is applied to a deducible sequent (in Intuitionistic Logic), it generates only deducible sequents (in Intuitionistic Logic).*

Proof. Suppose we have a deducible sequent as input. By applying the transformation rules, termination guarantees that eventually we'll get only sequents without any $\rightarrow$ connectives. But all these sequents, by the hypothesis, are deducible, and by theorem 5.4, they are abstractive. So we know how to compute their corresponding combinators and, by correctness, also a combinator corresponding to the input sequent. ∎

In the following, we present rules that perform equistructural transformations having in mind the conditions above. Formally, the equistructural transformation of a sequent generates a pair $\langle S, f\rangle$, where S is a set of sequents and f a function which allows us to compute the input sequent's corresponding combinator in terms of the corresponding combinators of the sequents in S.

6.1 Pure Transformation Rules

In a pure equistructural transformation, the input sequent is transformed into another sequent with the same combinator. This is represented by

$$\mathbf{X} * \Gamma \vdash \chi \stackrel{\text{ET}}{\Longrightarrow} \mathbf{X} * \Gamma_1 \vdash \chi_1.$$

ET1: Connective transformations. Connective rules do not refer to combinators, and therefore are obviously pure equistructural transformations. The rules we need are:

$$\begin{array}{ll} (a) & \mathbf{X} * \Gamma \vdash \varphi{\rightarrow}\chi \stackrel{ET1}{\Longrightarrow} \mathbf{X} * \Gamma, \varphi \vdash \chi. \\ (b) & \mathbf{X} * \Gamma[\varphi{\otimes}\psi] \vdash \chi \stackrel{ET1}{\Longrightarrow} \mathbf{X} * \Gamma[\varphi, \psi] \vdash \chi. \end{array}$$

Rule (a) is the converse of rule $(\vdash \rightarrow)$ and rule (b) is the converse of $(\otimes \vdash)$. Both rules are easily proven admissible (with the use of Cut). So ET1 satisfies the condition of Lemmas 6.3 and 6.5.

Notation simplification: In all examples below (and in some of the proof trees as well) we will omit the symbol $\otimes$, writing $\varphi\psi$ instead of $\varphi\otimes\psi$; this should cause no confusion because φ and ψ are always formulas.

Example 6.6 *Consider the deducible purely logical sequent*

$$\mathbf{X} \vdash p{\rightarrow}(q{\rightarrow}pq).$$

By applying ET1(a) twice we get

$$\mathbf{X}, p, q \vdash pq$$

which is a λ-abstractive sequent, so we have to find a combinator equivalent to $\lambda xy.xy$. But by Corollary 5.5(b), we can simplify this task, computing $\mathbf{X} = \lambda x.x = \mathbf{I}$.

ET2: Grouping Formulas. Let $\mathbf{X} * \Gamma \vdash \chi$ be a deducible purely logical sequent. We can group and substitute its formulas in two ways.

(a) If ψ occurs once or more in the sequent, such that none of its subformulas occur outside ψ, then ψ is substituted by a new atomic symbol α.

(b) If $\chi = \chi_1 \otimes \ldots \otimes \chi_n$ such that all χ_i occurs as a component in Γ, then each component in the sequent is substituted by a new symbol in the sequent; we obtain thus an abstractive sequent equistructural to the input one.

Those are clearly pure equistructural transformations that amount to manipulating substituted formulas as an atom. If φ is a formula being substituted above by a new symbol α, we can postulate that

$$\alpha \vdash \varphi \text{ and } \varphi \vdash \alpha$$

¿From this, with simple uses of the Cut rule, we show that the conditions of Lemmas 6.3 and 6.5 are satisfied.

Example 6.7 *Consider the deducible sequent* $\mathbf{X}, p{\rightarrow}q_1, q_2q_1, q_2{\rightarrow}r \vdash (p{\rightarrow}q_1)(q_2{\rightarrow}r)$. *We obtain,*

$$\begin{array}{ll} \mathbf{X}, \alpha_1, (q_2q_1), \alpha_2 \vdash \alpha_1\alpha_2 & \textit{by ET2(b)} \\ \mathbf{X}, \alpha_1, \alpha_3, \alpha_2 \vdash \alpha_1\alpha_2 & \textit{by ET2(a)} \end{array}$$

a λ-abstractive sequent leading to $\lambda xyz.xz$. By Corollary 5.5(b), we simplify it, computing $\mathbf{X} = \lambda xy.x = \mathbf{K}$.

6.2 Eliminating $\rightarrow$ from Antecedents

Pure transformations, especially ET1(a), help us eliminate succedents of the form $\varphi{\rightarrow}\psi$. We now transform a sequent to eliminate formulas of the form $\varphi{\rightarrow}\psi$ from the antecedent.

Let $\Gamma[\psi][\xi]$ represent an antecedent containing the formulas ψ and ξ as components (*i.e.* not inside an $\rightarrow$-formula); the order between ψ and ξ can be any. In this case, $\Gamma[\psi'][]$ represents the sequent obtained by substituting ψ with ψ' and deleting ξ.

ET3: ($\rightarrow \vdash$)-guided transformation. Consider a purely logical sequent $\mathbf{X} * \Gamma[\varphi{\rightarrow}\psi][\varphi] \vdash \chi$. This sequent is equistructurally transformed into:

$$S = \left\{ \begin{array}{ll} \text{(i)} & \mathbf{X}_1 * \Gamma[\psi, \varphi][] \vdash \chi \\ \text{(ii)} & \mathbf{X}_2\mathbf{X}_1 * \Gamma[\varphi{\rightarrow}\psi][\varphi] \vdash \mathbf{X}_1 * \Gamma[(\varphi{\rightarrow}\psi, \varphi, \varphi)][] \end{array} \right\} \quad \text{and } \mathbf{X} = \mathbf{X}_2\mathbf{X}_1$$

The formula $\varphi{\rightarrow}\psi$ is called the *ticket* formula, and the separate occurrence of ψ is called the *riding formula*. Assuming that the sequents in S are deducible, the correctness of the equistructural transformation above comes from:

$$\dfrac{\dfrac{\text{(i) } \mathbf{X}_1 * \Gamma[\psi, \varphi][] \vdash \chi \quad \varphi \vdash \varphi}{\mathbf{X}_1 * \Gamma[(\varphi{\rightarrow}\psi, \varphi, \varphi)][] \vdash \chi}(\rightarrow\vdash) \quad \text{(ii) } \mathbf{X}_2\mathbf{X}_1 * \Gamma[\varphi{\rightarrow}\psi][\varphi] \vdash \mathbf{X}_1 * \Gamma[(\varphi{\rightarrow}\psi, \varphi, \varphi)][]}{\mathbf{X}_2\mathbf{X}_1 * \Gamma[\varphi{\rightarrow}\psi][\varphi] \vdash \chi}\text{(Cut)}$$

The use of the Cut Rule above is merely the transitivity of $\vdash$ (module the use of ET1(b and c)). In sequent (ii), if by ET2(b) we substitute non-atomic formulas (and $\mathbf{X}_1$) by new symbols, it becomes an abstractive sequent, and therefore always deducible.

The reason that (i) is not simply of the form (i′) $\mathbf{X}_1 * \Gamma[\psi][] \vdash \chi$, as one could expect — it would certainly be a simpler sequent — is that we could not guarantee cc-completeness this way. In many cases, taking (i′) instead of (i) correctly computes a simpler combinator (see example 6.8 below); however, it does not always decompose a deducible sequent into a set of deducible sequents. A counterexample to the correct use of (i′) would be the deducible sequent

$$p{\to}q, p, p{\to}r \vdash qr$$

where ET3 is needed twice. If (i′) is used instead of (i), p will be deleted after the first application of ET3 we would get a non-deducible sequent.

As we presented it, the following derivation shows that the condition of Lemma 6.5 is observed in Intuitionistic Logic with the use of (i).

$$\dfrac{\dfrac{\Gamma[\varphi{\to}\psi][\varphi] \vdash \chi}{\Gamma[\varphi{\to}\psi, \varphi][] \vdash \chi}\;\text{(Structural manipulation)} \qquad \psi \vdash \varphi{\to}\psi}{\Gamma[\psi, \varphi][] \vdash \chi}\;\text{(Cut)}$$

The problem remains that (i) is guaranteed to generate too many duplications of the riding formula. For each superfluous use of (i) in the computation, a pair of combinators **W** (that duplicates the riding formula) and **K** (that later eliminates it) will be present in the computed combinator.

Using (i′), if it turns out that $\mathbf{X}_1 = \mathbf{I}$, the deduction of the input sequent may be simplified, for an ($\mathbf{I}\vdash$)-rule can always be substitute by the reflexivity-rule. Thus if $\Gamma[\psi][] \vdash \chi$ turns out to be of the form $\varphi \vdash \varphi$, then $\mathbf{X} = \mathbf{X}_2$ obtained from $\mathbf{X}_2 * \Gamma[\varphi{\to}\psi][\varphi] \vdash \Gamma[(\varphi{\to}\psi, \varphi)][]$. This simplification yields a "smaller" combinator as shown in the example below.

Example 6.8 *Consider the sequent* $\mathbf{X} \vdash (p{\to}q){\to}[(r{\to}p){\to}(r{\to}q)]$*; by ET1(a) it is equistructurally transformed into*

$$\mathbf{X}, p{\to}q, r{\to}p, r \vdash q$$

We now use ET3 with (i′), equistructurally transforming th sequent above into

$$\mathbf{X}_1, p{\to}q, p \vdash q$$
$$\mathbf{X}_2\mathbf{X}_1, p{\to}q, r{\to}p, r \vdash \mathbf{X}_1, p{\to}q, (r{\to}p, r)$$

with $\mathbf{X} = \mathbf{X}_2\mathbf{X}_1$*. The first sequent can be further transformed, still using (i′), into* $\mathbf{X}_3 q \vdash q$ *and* $\mathbf{X}_4\mathbf{X}_3, p{\to}q, q \vdash \mathbf{X}_3(p{\to}q, q)$*, with* $\mathbf{X}_1 = \mathbf{X}_4\mathbf{X}_3$*. Clearly,* $\mathbf{X}_3 = \mathbf{I}$ *and* $\mathbf{X}_4 = \mathbf{B}$*. Since* $\mathbf{X}_3 = \mathbf{I}$*, we simplify* $\mathbf{X}_1 = \mathbf{X}_4$ *and* $\mathbf{X}_4$ *is obtained from* $\mathbf{X}_4 p{\to}q, q \vdash p{\to}q, q$*.*

So it turns out that $\mathbf{X}_1 = \mathbf{I}$ *as well, allowing further simplifications; hence* $\mathbf{X} = \mathbf{X}_2$ *obtained from* $\mathbf{X}_2, p{\to}q, r{\to}p, r \vdash p{\to}q, (r{\to}p, r)$*, which yields* $\mathbf{X}_2 = \mathbf{B}$*. Finally, as expected,* $\mathbf{X} = \mathbf{B}$*.*

However, if we do follow the official rule and use (i) instead of (i′), no simplification is applicable and the final result is

$$\mathbf{X} = \mathbf{B}(\mathbf{BW})\left(\ \mathbf{BB}\left(\ \mathbf{BW}(\mathbf{B}(\mathbf{B}(\mathbf{BK})\mathbf{K}))\ \right)\ \right)$$

We can in fact construct derivations for both this combinator and **B** *as the corresponding combinators; the steps in their derivations tell us how. Just note that this* *last combinator has two pairs* **W**-**K**, *suggesting two superfluous uses of (i), as indeed is the case.*

Sometimes the sequent is in a format that ET3 cannot be applied directly. For those cases, we provide further transformations that will produce a sequent in a format suitable for ET3.

ET4: $\otimes$-guided transformation. We now consider the transformation that is guided by the presence of a component in the antecedent of the form $\Gamma[(\varphi_1\otimes\varphi_2)\to\psi]$; we indeed demand that the input sequent be of the form $\Gamma[(\varphi_1\otimes\ldots\otimes\varphi_m)\to\psi][\varphi_1]$, although this format information is not used in the transformation. We want to transform the ticket implication $(\varphi_1\otimes\varphi_2)\to\psi$ into a formula where ET3 can be applied, which is $\varphi_1\to(\varphi_2\to\psi)$. This is justified by the following derivation.

$$\cfrac{\cfrac{\cfrac{\varphi_1\varphi_2\to\psi \vdash \varphi_1\varphi_2\to\psi}{\varphi_1\varphi_2\to\psi, (\varphi_1\varphi_2) \vdash \psi}\;(\to\vdash)}{\mathbf{B}, \varphi_1\varphi_2\to\psi, \varphi_1, \varphi_2 \vdash \psi}\;(\mathbf{B}\vdash)}{\mathbf{B}, \varphi_1\varphi_2\to\psi \vdash \varphi_1\to(\varphi_2\to\psi)}\;(\to\vdash)$$

We can then use this derivation to transform a sequent of the form $\Gamma[(\varphi_1\otimes\varphi_2)\to\psi] \vdash \chi$:

$$\cfrac{\cfrac{\text{(i) } \mathbf{X}_1 * \Gamma[\varphi_1\to(\varphi_2\to\psi)] \vdash \chi \quad \mathbf{B}, \varphi_1\varphi_2\to\psi \vdash \varphi_1\to(\varphi_2\to\psi)}{\mathbf{X}_1 * \Gamma[\mathbf{B}(\varphi_1\varphi_2\to\psi)] \vdash \chi} \quad \text{(ii) } \mathbf{X}_2\mathbf{B}\mathbf{X}_1 * \Gamma[\varphi_1\varphi_2\to\psi] \vdash \mathbf{X}_1 * \Gamma[\mathbf{B}(\varphi_1\varphi_2\to\psi)]}{\mathbf{X}_2\mathbf{B}\mathbf{X}_1 * \Gamma[\varphi_1\varphi_2\to\psi] \vdash \chi}$$

Both steps in the deduction above are Cut steps. Based on that, we define the *equistructural $\otimes$-guided transformation* as taking a sequent of the form $\mathbf{X}*\Gamma[\varphi_1\otimes\varphi_2\to\psi] \vdash \chi$ and transforming it into:

$$S = \left\{ \begin{array}{ll} \text{(i)} & \mathbf{X}_1 * \Gamma[\varphi_1\to(\varphi_2\to\psi)] \vdash \chi \\ \text{(ii)} & \mathbf{X}_2\mathbf{B}\mathbf{X}_1 * \Gamma[\varphi_1\varphi_2\to\psi] \vdash \mathbf{X}_1 * \Gamma[\mathbf{B}(\varphi_1\varphi_2\to\psi)] \end{array} \right\} \quad \text{and } \mathbf{X} = \mathbf{X}_2\mathbf{B}\mathbf{X}_1$$

Note that (ii), as before, can be equistructurally transformed into an abstractive sequent by means of ET2(b), and thus $\mathbf{X}_2$ is computed. Sequent (i) is now ready for the application of transformation ET3. The condition of Lemma 6.3 is satisfied.

To show that ET4 also satisfies the condition of Lemma 6.5, we show that (i) can be obtained from the input formula; note that $\varphi_1\to(\varphi_2\to\psi) \vdash \varphi_1\varphi_2\to\psi$ is deducible in Intuitionistic Logic, so

$$\cfrac{\Gamma[\varphi_1\varphi_2\to\psi] \vdash \chi \quad \varphi_1\to(\varphi_2\to\psi) \vdash \varphi_1\varphi_2\to\psi}{\Gamma[\varphi_1\to(\varphi_2\to\psi)] \vdash \chi}\;(\text{Cut})$$

is a deduction of (i). As for (ii), as in the previous case, it is treated as a simple structural manipulation of an axiom, and so it is always deducible.

In a sequence of transformations, ET4 is always used in connection with some use of ET3; so its use is either [ET4;ET3] or [ET4;. . .;ET4;ET3] (in case $\varphi_1 = \psi_1\otimes\ldots\otimes\varphi_m$).

ET5: $\to$-guided transformation. Consider an antecedent Γ with a component of the form $\varphi_1\to(\varphi_2\to\ldots(\varphi_m\to\varphi)\ldots)\to\psi$ such that φ is also a component of Γ but $\varphi_1\to(\varphi_2\to\ldots(\varphi_m\to\varphi)\ldots)$ is not (otherwise we could apply ET3).

What we want to transform here is the riding formula φ, for if we can transform it into $\varphi_1{\rightarrow}(\varphi_2{\rightarrow}\ldots(\varphi_m{\rightarrow}\varphi)\ldots)$, ET3 becomes applicable. However, as with ET3, if the transformation deletes φ, the transformation may generate formulas that are not deducible; so the riding formula φ has to be preserved (duplicated) as well as transformed. First, we define the following abbreviation:

$$\begin{aligned}&\mathbf{K}^{[1]} = \mathbf{K}\\ &\mathbf{K}^{[i+1]} = \mathbf{BKK}^{[i]}\end{aligned}$$

It is easy to see that $\mathbf{K}^{[m]}xx_1\ldots x_m \rhd x$, from which we conclude that the sequent $\mathbf{K}^{[m]}\varphi \vdash \varphi_1{\rightarrow}(\ldots(\varphi_m{\rightarrow}\varphi)\ldots)$ is deducible for $m \geq 1$. The transformation of $\mathbf{X} * \Gamma[\varphi] \vdash \chi$ follows from the following deduction:

$$\dfrac{\dfrac{\text{(i) } \mathbf{X}_1 * \Gamma[\varphi_1{\rightarrow}(\ldots(\varphi_m{\rightarrow}\varphi)\ldots),\varphi] \vdash \chi \quad \mathbf{K}^{[m]}\varphi \vdash \varphi_1{\rightarrow}(\ldots(\varphi_m{\rightarrow}\varphi)\ldots)}{\mathbf{X}_1 * \Gamma[\mathbf{K}^{[m]}\varphi,\varphi] \vdash \chi} \quad \text{(ii) } \mathbf{X}_2\mathbf{K}^{[m]}\mathbf{X}_1 * \Gamma[\varphi] \vdash \mathbf{X}_1 * \Gamma[\mathbf{K}^{[m]}\varphi,\varphi]}{\mathbf{X}_2\mathbf{K}^{[m]}\mathbf{X}_1 * \Gamma[\varphi] \vdash \chi}$$

where both steps above are applications of the Cut-rule.

It follows that the *equistructural* $\rightarrow$*-transformation* takes a sequent of the form $\mathbf{X} * \Gamma[\varphi] \vdash \chi$ and transforms it into:

$$S = \left\{ \begin{array}{ll} \text{(i)} & \mathbf{X}_1 * \Gamma[\varphi_1{\rightarrow}\ldots(\varphi_m{\rightarrow}\varphi)\ldots,\varphi)] \vdash \chi \\ \text{(ii)} & \mathbf{X}_2\mathbf{K}^{[m]}\mathbf{X}_1 * \Gamma[\varphi] \vdash \mathbf{X}_1 * \Gamma[\mathbf{K}^{[m]}\varphi] \end{array} \right\} \qquad \text{and } \mathbf{X} = \mathbf{X}_2\mathbf{K}^{[m]}\mathbf{X}_1$$

This transformation satisfies the conditions of Lemma 6.3. That it also satisfies the conditions of Lemma 6.5 comes from:

$$\dfrac{\Gamma[\varphi] \vdash \chi}{\Gamma[\psi{\rightarrow}\varphi,\varphi] \vdash \chi}\ \text{(Thinning)}$$

The computation of $\mathbf{X}_2$ follows the same steps as before. In a sequence of transformations, ET5 is always used in connection with some use of ET3; so its use is either [ET5;ET3] or [ET5;ET4;. . .;ET4;ET3] (in case $\varphi = \psi_1\otimes\ldots\otimes\psi_m$).

6.3 The Split Transformation

In all transformations above, combinator $\mathbf{X}_2$ in sequent (ii) can be computed with the same steps, namely, an application of ET2(b) that will transform (ii) into an abstractive sequent, followed by complex combinator abstraction. So it is sequent (i) that, in all cases, will be subject to further transformations and transformations. Successive applications of ET1–ET5 will eliminate the occurrences of some $\rightarrow$-connectives from the antecedent, but maybe not all of them; however, ET1 guarantees that the succedent is not of the form $\varphi{\rightarrow}\psi$. We reach a point where ET1–ET5 are not applicable and the succedent is either atomic, or a conjunction of atoms, or a conjunction containing some $\rightarrow$-formula.

In the first two cases the succedent is in the $\otimes$-fragment, and we know how to decide if it is deducible and how to compute its corresponding combinator.

So we are faced with the case in which the succedent is of the form $\chi_1 \otimes \chi_2$ such that χ_1 or χ_2 contains an $\rightarrow$ connective. So we split the input sequent in three other sequents.

ET6: Split transformation. Consider a sequent of the form $\mathbf{X} * \Gamma \vdash \chi_1 \otimes \chi_2$, where ET1–ET5 are not applicable and χ_1 or χ_2 contains an $\rightarrow$ connective. This sequent is split into 3 sequents:

$$S = \left\{ \begin{array}{ll} (i) & \mathbf{X}_1 * \Gamma \vdash \chi_1 \\ (ii) & \mathbf{X}_2 * \Gamma \vdash \chi_2 \\ (iii) & \mathbf{X}_3\mathbf{X}_1\mathbf{X}_2 * \Gamma \vdash (\mathbf{X}_1 * \Gamma) \otimes (\mathbf{X}_2 * \Gamma) \end{array} \right\}$$
$$\text{and } \mathbf{X} = \mathbf{X}_3\mathbf{X}_1\mathbf{X}_2$$

The third sequent is responsible for recomposing the split sequents, as shown in the following deduction:

$$\dfrac{\dfrac{(\mathbf{X}_1 * \Gamma \vdash \chi_1) \quad (\mathbf{X}_2 * \Gamma \vdash \chi_2)}{(\mathbf{X}_1 * \Gamma), (\mathbf{X}_2 * \Gamma) \vdash \chi_1 \otimes \chi_2}\,(\vdash \otimes) \qquad \mathbf{X}_3\mathbf{X}_1\mathbf{X}_2 * \Gamma \vdash (\mathbf{X}_1 * \Gamma) \otimes (\mathbf{X}_2 * \Gamma)}{\mathbf{X}_3\mathbf{X}_1\mathbf{X}_2 * \Gamma \vdash \chi_1 \otimes \chi_2}\,(\text{Cut})$$

$\mathbf{X}_3$ is computed from sequents (i) and (ii) using transformation ET2(b). $\mathbf{X}_1$ and $\mathbf{X}_2$ are computed with further applications or decomposition rules. The conditions of correctness are clearly satisfied by ET6. To show that the conditions of

6.4 Guaranteeing $\rightarrow$-elimination from the sequent

At this point we are guaranteed to eliminate all the $\rightarrow$ connectives from the succedent, which is a formula in the $\otimes$-fragment (see proof of Theorem 6.10 below). We claim that in such a case (see Lemma 6.9) the $\rightarrow$-formula could only have been introduced in the deduction by the rule $(\mathbf{K} \vdash)$, or in a Intuitionistic Deduction, by Thinning.

Therefore we propose the substitution of $\rightarrow$-formula of the form $\varphi \rightarrow \psi$ by a new atomic symbol α, meaning that the formula $\varphi \rightarrow \psi$ could have been manipulated as an atomic symbol.

ET7: Elimination transformation. We are faced with a situation in which the antecedent contains a formula $\varphi \rightarrow \psi$ and none of ET1–ET6 can be applied. We then introduce a new symbol α and make it identical to $\varphi \rightarrow \psi$, *i.e.*

$$\alpha \vdash \varphi \rightarrow \psi \quad \text{and} \quad \varphi \rightarrow \psi \vdash \alpha$$

We can then substitute $\varphi \rightarrow \psi$ by α. The correctness of this step is justified by the following derivation.

$$\dfrac{{\scriptstyle (i)}\ \mathbf{X}_1 * \Gamma[\alpha] \vdash \chi \quad \varphi \rightarrow \psi \vdash \alpha}{\mathbf{X}_1 * \Gamma[\varphi \rightarrow \psi] \vdash \chi}\,(\text{Cut})$$

So, the *equistructural elimination transformation* can be seen as a *pure* transformation that takes a sequent of the form $\mathbf{X} * \Gamma[\varphi \rightarrow \psi] \vdash \chi$ and transforms it into $\mathbf{X} * \Gamma[\alpha] \vdash \chi$, for α a new atomic symbol.

The combinator $\mathbf{K}$ will necessarily be in $\mathbf{X}$, representing the elimination of a formula from the antecedent (or the use of thinning in the deduction). The justification for this fact, which also shows that this transformation fulfills the conditions of Lemma 6.5 comes from the following result.

Lemma 6.9 *Let $\Gamma[\varphi{\rightarrow}\psi] \vdash \chi$ be a deducible sequent such that χ is $\rightarrow$-free. If the inference rule $(\rightarrow\vdash)$ is used in any branch of its deduction, then either ET3 or ET4 are applicable at $\Gamma[\varphi{\rightarrow}\psi] \vdash \chi$.*

Proof. By induction on the number of inference rules applications separating the use of $(\rightarrow\vdash)$ and $\Gamma[\varphi{\rightarrow}\psi] \vdash \chi$. We first note that no inference rule removes a formula from the succedent, so we can assume that succedents are $\rightarrow$-free throughout the deduction.

The base case is right after the use of $(\rightarrow\vdash)$, at which point ET1 is clearly applicable.

Suppose we have an intermediate sequent in the branch of the form $\Delta[\varphi{\rightarrow}\psi] \vdash \xi$. By induction hypothesis, either ET3 or ET4 are applicable. Note that rules $(\otimes\vdash)$, $(\vdash\otimes)$ and $(\rightarrow\vdash)$ do not affect the applicability of ET3 and ET4. Rule $(\vdash\rightarrow)$ cannot be applied because the succedent is $\rightarrow$-free at all deductive steps.

Finally, we have to consider structural rules (or combinator rules); but all they can do is to move $\varphi{\rightarrow}\psi$ and φ around, and in case φ is of the form $\varphi_1{\otimes}\varphi_2$, split it. So the applicability of ET3 can be changed to the applicability of ET4 and vice-versa. But at the next sequent in the branch, it remains the case that either ET3 or ET4 are applicable. ∎

There are only two ways in which an $\rightarrow$-formula may be inserted in the antecedent of a sequent: either via rule $(\rightarrow\vdash)$ and via rule $(\mathbf{K}\vdash)$. If the rules ET1–ET6 are not applicable, Lemma 6.9 yields that in the deduction of $\Gamma[\varphi{\rightarrow}\psi] \vdash \chi$ rule $(\rightarrow\vdash)$ could not have been used, so the $\rightarrow$-formula $\varphi{\rightarrow}\psi$ was introduced with $(\mathbf{K}\vdash)$, and $\varphi{\rightarrow}\psi$ was manipulated as an atomic formula. From such an argumentation we deduce from

$$\Gamma[\varphi{\rightarrow}\psi] \vdash \chi$$

the fact that

$$\Gamma[\alpha] \vdash \chi$$

by substituting the insertion of $\varphi{\rightarrow}\psi$ by the rule $(\mathbf{K}\vdash)$, with an insertion of α. So the condition for cc-completeness in Lemma 6.5 is met.

6.5 Termination, Correctness and cc-Completeness

Theorem 6.10 *Algorithm 6.1 always terminates.*

Proof. We note that each sequent generated by an equistructural transformation has fewer connectives than the transformed sequent. This guarantees that, eventually either we get stuck, without a transformation to apply, or we get all sequents in the $\otimes$-fragment, where by Theorem 5.4 we know how to finish the computation.

We claim that the algorithm never gets stuck. Indeed, suppose we get stuck; then there is a sequent with an $\rightarrow$ connective in it, otherwise all sequents are in the $\otimes$-fragment and the algorithm finishes. But the $\rightarrow$-formula cannot be in the succedent because:

- if the succedent is of the form $\varphi{\rightarrow}\psi$, ET1(a) is applicable.

- if the succedent is of the form $\chi_1 \otimes \chi_2$ and the $\rightarrow$-formula is contained in χ_1 or χ_2, ET6 is applicable.

Both cases contradict the fact that the algorithm got stuck.

So we have a succedent in the $\otimes$-fragment, and a $\rightarrow$-formula in the antecedent and the algorithm is stuck. Due to rule ET1(b), we know the antecedent is of the form $\Gamma[\varphi \rightarrow \psi] \vdash \chi$. There are four rules to be potentially applicable: ET3, ET4, ET5 and ET7. If the first three rules —ET3, ET4 and ET5— are not applicable, then necessarily ET7 will be applicable (and only in such a case). So the algorithm does not get stuck when there is an $\rightarrow$-formula in the antecedent.

Therefore Algorithm 6.1 always terminates. ∎

Theorem 6.11 *Algorithm 6.1 is correct and cc-complete.*

Proof. Just note that all rules above satisfy the sufficient conditions imposed by Lemmas 6.3 and 6.5. ∎

Corollary 6.12 *Algorithm 6.1 can be used to decide the* $\{\otimes, \rightarrow\}$*-fragment of Intuitionistic Logic.*

Proof. Straight from the combination of Theorem 6.11 and Theorem 4.5. ∎

7 Conclusions and Further Work

In this paper we have shown that it is possible to automatically infer a family of substructural logics in which a given formula/sequent is deducible.

The main deficiency of the algorithm proposed in Section 6 is that we do not always find the family of all logics in which a formula is deducible. One possible way to do that would be to change our transformation rules so that our algorithm could compute a *minimal* combinator in some sense. If this minimal combinator were unique, it would represent the desired family of all logics in which a formula is deducible. However, as Example 6.8 shows, the combinators generated are not minimal in any useful sense. Indeed, for the sake of obtaining cc-completeness, transformation rule ET3 causes the generation of combinators that are "too big". It would be interesting to investigate refinements of ET3 that would not only preserve cc-completeness but also guarantees minimality (and thus strong completeness. A proper notion of order for complex combinators also needs to be developed.

We would also like to extend our approach to richer fragments, including negation and the additive connectives. Here we may reach a decidability threshold quite easily and it would be interesting to investigate which fragments still permit a terminating algorithm to compute an associated combinator. Obviously, we must restrict ourselves to decidable fragments.

Finally, the implementation of structurally-free theorem provers is certainly a topic worth of further investigation that we wish to pursue.

Acknowledgments

This work was partly supported by the Brazilian Research Council (CNPq), grant 300597/95-9. We would like to thank a reviewer for helpful comments.

References

[1] A. R. Anderson and N. D. Belnap Jr. *Entailment: The Logic of Relevance and Necessity*, volume 1. Princeton University Press, 1975.

[2] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Number 103 in North-Holland Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers, 1981.

[3] K. Broda, M. Finger, and A. Russo. LDS-Natural Deduction for Substructural Logics. In *3rd Workshop on Logic, Language, Information and Computation (WoLLIC96)*, pages 15–18, 1996. Abstract only.

[4] M. W. V. Bunder. *Set Theory based on Combinatory Logic.* PhD thesis, University of Amsterdam, 1969.

[5] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematical Studies*, 33(2):346–366, 1932.

[6] Alonzo Church. A set of postulates for the foundation of logic (cont.). *Annals of Mathematical Studies*, 34:839–864, 1933.

[7] H. B. Curry, R. Feys, and W. Craig. *Combinatory Logics*, volume 1. North Holland, 1958.

[8] M. D'Agostino and D. Gabbay. A Generalization of Analithic Deduction via Labelled Tableaux, part I: Basic Substructural Logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

[9] Dirk Van Dalen. Intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III, 1984.

[10] K. Došen. A Historical Introduction to Substructural Logics. In P. S. Heister and K. Došen, editors, *Substructural Logics*, pages 1–31. Oxford University Press, 1993.

[11] J. M. Dunn. Proof Theory and Semantics for Structurally Free Logics. In *Proceedings of the 3rd Workshop of Logic, Language, Information and Computation (WoLLIC96)*, 1996. Abstract only.

[12] J. M. Dunn and R. K. Meyer. Combinators and Structurally Free Logic. *Logic Journal of the IGPL*, 5(4):505–538, July 1997.

[13] M. Finger. Computing List Combinator Solutions for Structural Equations. Technical Report RT-MAC-9711, Department of Computer Science, University of São Paulo, 1997. Available for ftp[4].

[14] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[15] J. R. Hindley, B. Lercher, and J. P. Seldin. *Introduction to Combinatory Logic.* Cambridge University Press, 1972.

[16] J. R. Hindley and J. P. Seldin. *Introduction to Combinators and the λ-Calculus.* Cambridge University Press, 1986.

[17] W. Howard. The formulae-as-types notion of construction. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[18] G. Revesz. *Lambda Calculus, Combinatorics and Functional Programming.* Number 4 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[19] A. Schönfinkel. Über die Bausteine der Mathematischen Logik. In J. van Heijennort, editor, *From Frege to Gödel.* Harvard University Press, Cambridge, Massachussets, 1967. Reprinted from original version, 1924.

Received February 10, 1998

[4] ftp://ftp.ime.usp.br/pub/mfinger/rtmac9711.ps.gz