$\mu\prod\lambda\forall$

Graduate Program in Logic, Algorithms and Computation

Normalisation in Deep Inference

Panos Tsatsanis

Supervisors:
Yiorgos Stavrinos George Koletsos

October 2011

Abstract

In this thesis we present the calculus of structures, a proof-theoretic formalism using deep inference. This means that inference rules apply arbitrarily deep inside formulas. It follows that derivations are now symmetric instead of tree-shape objects. A system for classical predicate logic is introduced and compared with the corresponding sequent calculus system. They both have an admissible Cut rule. However, locality can be obtained with deep inference, meaning that the effort of applying a rule is always bounded. Then we investigate what normal forms of deductions have been defined. Besides cut elimination, we can adopt two other notions of normalisation that allow cuts inside a derivation, under some constraints. We will try to remark common things and differences between normalisation in deep and shallow inference.

Acknowledgements

I would like to thank the S.I.C. of MPLA for giving me the opportunity to become a student of this program. I want to express my deepest gratitude to everyone involved in MPLA for the great amount of knowledge that was offered to me.

In particular, I want to thank my thesis advisor Yiorgos Stavrinos, for his patient guidance and advice throughout our systematic work. He has been a great teacher for me, both with his lectures in the courses and during the writing of this thesis.

I should then mention Costas Dimitracopoulos, who keeps logic in very high level in MPLA. He has taught me a lot more than he had to. I certainly have to thank George Koletsos, as the current chairman of MPLA but mainly as the person who introduced me to logic during my undergraduate studies in NTUA. I would also like to thank Panos Rondogiannis for his inspiring course on logic programming semantics.

Contents

List of Figures

Chapter 1

Introduction

A **logic** in a given formal language consists of a class of **valid** sentences. These will be the formulas of the language considered always true in that logic, according to some kind of semantics. Mathematics is usually performed in **first-order classical logic**. Classical logic is based on the standard language of logic where $\top, \bot$ denote the truth values, $\vee, \wedge, \rightarrow, \neg$ stand for disjunction, conjunction, implication and negation, and $\exists, \forall$ are the existential and the universal quantifier. Classical logic contains all the sentences considered as tautologies in usual mathematics. By first-order we mean that we use quantifiers only on elements of the universe and not on subsets of it. Sometimes we use weaker logics, such as **intuitionistic** logic. These are logics in the same language but their class of theorems is a subset of classical logic. For example, in intuitionistic logic the **law of the excluded middle** $A \vee \neg A$ does not hold. Logics contained in classical logic and containing intuitionistic logic are called **intermediate**. There are also many logics far apart from classical logic, like **modal** logics. They extend the language of classical logic with special connectives, operators etc.

A **proof system** for a given logic is a set of syntactic inference rules such that: a sentence is true in that logic if and only if there is a sequence of rules, i.e. a **formal proof**, that ends with that sentence as conclusion. Constructing a formal proof is a mechanical procedure. Computers can be employed for checking and discovering proofs.

Proof systems can be classified according to the style they adhere to. Those styles are called **formalisms**. One of the main formalisms in proof theory is the **sequent calculus**, introduced by Gerhard Gentzen in 1934. It is based on a syntactic expression called **sequent**, which is of the following shape:

$$B_1, ..., B_m \vdash A_1, ..., A_n$$

where $A_1, ..., A_n, B_1, ..., B_m$ are formulas. The formulas on the left-hand side of the **turnstile** $\vdash$ are called the **antecedent**, and the formulas on the right-hand side are called the **succedent**. The sequent above is interpreted as:

$$(B_1 \text{ and } ... \text{ and } B_m) \text{ implies } (A_1 \text{ or } ... \text{ or } A_n)$$

The sequent system G1c for classical logic can be seen in Figure 1.1. The letter G stands of course for "Gentzen" and c for "classical". G1c is a variant of the original Gentzen's system LK. The rules are devided into **left** and **right** introduction rules. Rules $\wedge_L, \wedge_R, \vee_L, \vee_R, \rightarrow_L, \rightarrow_R$ are the **logical** rules of the system, they introduce a logical connective. Rules w_L,w_R,c_L,c_R, along with **commutativity**:

$$A \vee B \leftrightarrow B \vee A$$
$$A \wedge B \leftrightarrow B \wedge A$$

and **associativity**:

$$(A \vee B) \vee C \leftrightarrow A \vee (B \vee C)$$
$$(A \wedge B) \wedge C \leftrightarrow A \wedge (B \wedge C)$$

are the **structural** rules. The w stands for **weakening** and c for **contraction**. There are logics that are characterised by the absence of some of those rules. They are called **substructural** logics. As an example we can mention non-commutative logics or **linear logic**, which does not allow weakening and contraction. The greek capitals, such as Φ and Ψ, denote multisets of formulas. In the rules they play the role of the **context**, i.e. formulas that do not participate in the inference. The formulas in the premisses that do not belong to the context are the **active** formulas. Negation $\neg A$ is defined as $A \rightarrow \bot$. The propositional system G1cp consists of the same rules except $\forall_I, \forall_E, \exists_I, \exists_E$.

Deductions, also called **derivations**, are trees of the following form:

$$\begin{array}{c} \Gamma_1 \vdash \Delta_1 \quad \cdots \quad \Gamma_n \vdash \Delta_n \\ \bigtriangledown \\ \Gamma \vdash \Delta \end{array}$$

where $\Gamma_1 \vdash \Delta_1, ..., \Gamma_n \vdash \Delta_n$ are the **premisses** and $\Gamma \vdash \Delta$ the **conclusion**. If all leaves are axioms or $\top$ instances, instead of arbitrary sequents, we call it a **proof**. For example, what follows below is the proof of the sequent $A \to (A \to B) \vdash (A \to B)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,\text{Ax}}{A, B \vdash B}\,\text{W}_L \qquad \dfrac{}{A \vdash A}\,\text{Ax}}{A, A \to B \vdash B}\,{\to_L} \qquad \dfrac{}{A \vdash A}\,\text{Ax}}{A, A \to (A \to B) \vdash B}\,{\to_L}}{A \to (A \to B) \vdash (A \to B)}\,{\to_R}$$

G1c is **sound** with respect to classical logic, i.e. provable formulas are valid. It is also **complete** with respect to classical logic, which means that if a formula is valid then it is provable. Thus valid sentences are exactly those formulas A where $\vdash A$ is a provable sequent. A valid formula is proved under no hypotheses.

The system comes with the **Cut** rule:

$$\text{Cut}\,\frac{\Phi \vdash \Psi, A \qquad A, \Phi' \vdash \Psi'}{\Phi, \Phi' \vdash \Psi, \Psi'}$$

An application of the rule is called a **cut**. The formula that instantiates A is the **cut formula**. The Cut rule is **admissible**. This means that for every proof of $\Phi \vdash \Psi$ in G1c + Cut there is a proof of the same sequent in G1c, that is without cuts. Gentzen showed the admissibility by describing a syntactic algorithmic procedure which eliminates the cuts from any given proof. We will see it in Chapter 4.

Proofs in the sequent calculus are top-down asymmetric objects. They are trees with a single root and many leaves. Their tree-shape is due to the presence of two-premise inference rules. For example, in the $\wedge_R$ rule:

$$\wedge_R\,\frac{\Phi \vdash A, \Psi \qquad \Phi' \vdash B, \Psi'}{\Phi, \Phi' \vdash A \wedge B, \Psi, \Psi'}$$

$$\mathrm{Ax}\ \frac{}{A \vdash A} \qquad \bot\ \frac{}{\bot \vdash}$$

$$\mathrm{c}_L\ \frac{\Phi, A, A \vdash \Psi}{\Phi, A \vdash \Psi} \qquad \mathrm{c}_R\ \frac{\Phi \vdash A, A, \Psi}{\Phi \vdash A, \Psi}$$

$$\mathrm{w}_L\ \frac{\Phi \vdash \Psi}{\Phi, A \vdash \Psi} \qquad \mathrm{w}_R\ \frac{\Phi \vdash \Psi}{\Phi \vdash A, \Psi}$$

$$\wedge_L\ \frac{\Phi, A, B \vdash \Psi}{\Phi, A \wedge B \vdash \Psi} \qquad \wedge_R\ \frac{\Phi \vdash A, \Psi \qquad \Phi' \vdash B, \Psi'}{\Phi, \Phi' \vdash A \wedge B, \Psi, \Psi'}$$

$$\vee_L\ \frac{A, \Phi \vdash \Psi \qquad B, \Phi' \vdash \Psi'}{\Phi, \Phi', A \vee B \vdash \Psi, \Psi'} \qquad \vee_R\ \frac{\Phi \vdash A, B, \Psi}{\Phi \vdash A \vee B, \Psi}$$

$$\rightarrow_L\ \frac{A, \Phi \vdash A, \Psi \qquad \Phi', B \vdash \Psi'}{\Phi, \Phi', A \rightarrow B \vdash \Psi, \Psi'} \qquad \rightarrow_R\ \frac{A, \Phi \vdash \Psi, B}{\Phi \vdash \Psi, A \rightarrow B}$$

$$\forall_L\ \frac{\Phi, A[x/t] \vdash \Psi}{\Phi, \forall x A \vdash \Psi} \qquad \forall_R\ \frac{\Phi \vdash A[x/y], \Psi}{\Phi \vdash \forall x A, \Psi}\ y \notin FV\{\Phi, \Psi, \forall x A\}$$

$$\exists_L\ \frac{\Phi, A[x/y] \vdash \Psi}{\Phi, \exists x A \vdash \Psi}\ y \notin FV\{\Phi, \Psi, \exists x A\} \qquad \exists_R\ \frac{\Phi \vdash A[x/\tau], \Psi}{\Phi \vdash \exists x A, \Psi}$$

Figure 1.1: System G1c

there is an asymmetry between premise and conclusion. We have two premisses, but one conclusion. Suppose that we want somehow to fix this and obtain top-down symmetric proofs. One could think that a two-premise rule can be written equivalently with only one premise. For example, consider the $\vee_L$ rule:

$$\vee_L \frac{A, \Phi \vdash \Psi \qquad B, \Phi' \vdash \Psi'}{\Phi, \Phi', A \vee B \vdash \Psi, \Psi'}$$

If we identify the logical level with the structural level of the sequent calculus (Figure 1.2), the rule can be written:

$$\vee_L \frac{(A \wedge \Phi \to \Psi) \wedge (B \wedge \Phi' \to \Psi')}{(A \vee B) \wedge \Phi \wedge \Phi' \to \Psi \vee \Psi'}$$

However, dropping the structural connectives of the sequent calculus would render the system incomplete because the tree-shape of the rules is necessary in order to access subformulas. Consider the following derivation as an example:

$$\dfrac{\dfrac{\dfrac{B \vdash C}{\vdash B \to C}\,{\to_R} \qquad \vdash A}{\vdash A \wedge (B \to C)}\,{\wedge_R}}$$

Seen bottom-up, the reason why the $\to_R$ rule can eventually be applied to $B \to C$ is that the $\wedge_R$ rule below decomposes the conclusion and distributes its contents among the leaves of the derivation. If $\wedge_R$ was written in one-premise form this wouldn't happen.

Stuctural Level	Logical Level
Comma in the Antecedent	$\wedge$
Comma in the Succedent	$\vee$
Branching	$\wedge$
Turnstile ($\vdash$)	$\to$

Figure 1.2: Structural Connectives

Therefore, if we drop the tree-shape and the distinction between logical and structural level, we need to restore the ability of accessing subformulas. This is where **deep inference** comes into the picture. By this we mean that we will allow inference rules to apply anywhere deep inside a formula, not only at the main connective. Most of traditional proof theory adopts a methodology that we call **shallow inference**. Shallow inference rules operate on connectives that appear in close proximity to the root of formulas. For example, consider the $\rightarrow_R$ rule of G1c:

$$\rightarrow_R \frac{A, \Phi \vdash \Psi, B}{\Phi \vdash \Psi, A \rightarrow B}$$

Even if A had an internal structure, e.g. $A = B \wedge C$, we would not have access to it before removing the root connective $\rightarrow$. If we adopt deep inference there is equal access to every subformula of a formula, regardless its root. Then we can drop the structural connectives and observe **symmetry**. Inference rules are top-down symmetric objects, one premise and one conclusion. This symmetry of inference rules extends of course to derivations. All this effort is not motivated only by aesthetic considerations. Symmetry makes clear the duality between axioms and cuts. Furthermore, it allows dualising a derivation by negating everything and flipping it upside-down.

There are various formalisms using deep inference. In this thesis we will present the calculus of structures, a formalism which employs deep inference and symmetry. We will see a proof theory for classical logic much in the same way as in the sequent calculus, and also some new proof-theoretical properties. We will focus on the normalisation techniques.

Summary

Chapter 2 is an introduction to the calculus of structures. We will give the basic definitions for this formalism and study a deep inference system for classical logic. We will show that it is equivalent with a sequent system. It occurs that the cut rule of the calculus of structures is admissible as well.

In Chapter 3 we will see the notion of locality. The application of a local rule only affects a bounded portion of the formula it is applied to. Several rules will be replaced by their local versions. A local propositional system is presented and we show that this is impossible in sequent systems.

Chapter 4 deals with normalisation. In the sequent calculus a normal proof is a cut-free proof. We present three approaches of normalisation in the calculus of structures. In the first, a normal proof is a cut-free proof, as in sequent systems. Then we define what a normal derivation is. It is characterised by a certain decomposition of inference rules. At last, we give an alternative characterisation of a normal proof which does not demand cut elimination.

Chapter 2

Calculus of Structures

In this chapter we will study the properties of a deep inference formalism that we call the **calculus of structures (CoS)**. This is the simplest but the most central formalism in deep inference. Then we will present system SKSgq, a deductive system for classical predicate logic.

Our basis will be the sequent system GS1, shown in Figure 2.1. This is an **one-sided** system, since it uses sequents $\vdash A_1, ..., A_n$ instead of the more general, two-sided version $B_1, ..., B_m \vdash A_1, ..., A_n$. Such systems are called **Gentzen-Schütte** systems, this is what GS stands for. They do not exist for weaker logics. We are able to formulate them because of the **symmetry of classical logic**. By this we mean the tautology:

$$\neg\neg A \leftrightarrow A$$

also referred to as **law of double negation**. The implication from right to left holds intuitionistically but the other direction $\neg\neg A \rightarrow A$ is valid only in classical logic. An immediate consequence is the equivalence:

$$A \rightarrow B \leftrightarrow \neg A \vee B$$

This allows us to translate every sequent $A \vdash B$ to its one-sided version $\vdash \neg A, B$. It also explains why there is no rule for implication. It can be expressed by disjunction. Keep in mind that the law of the excluded middle, the symmetry of negation and other theorems that are valid only clasically are equivalent with the scheme of **proof by contradiction**: say that we wish to prove proposition p. We can proceed by assuming $\neg p$, and showing that it leads to a logical contradiction. Thus, $\neg p$ must be false, and p is true. In intuitionistic logic we can't use this technique, only straight proofs are allowed. Logics of that kind are called **constructive**.

$$\top \frac{}{\vdash \top} \qquad \mathrm{Ax} \frac{}{\vdash A, \bar{A}}$$

$$\wedge_R \frac{\vdash \Phi, A \qquad \vdash \Psi, B}{\vdash \Phi, \Psi, A \wedge B} \qquad \vee_R \frac{\vdash \Phi, A, B}{\vdash \Phi, A \vee B}$$

$$\mathrm{c}_R \frac{\vdash \Phi, A, A}{\vdash \Phi, A} \qquad \mathrm{w}_R \frac{\vdash \Phi}{\vdash \Phi, A}$$

$$\exists_R \frac{\vdash \Phi, A[x/\tau]}{\vdash \Phi, \exists x A} \qquad \forall_R \frac{\vdash \Phi, A[x/y]}{\vdash \Phi, \forall x A} \quad y \notin FV\{\Phi, \forall x A\}$$

Figure 2.1: System GS1

The use of one-sided sequents halves the number of rules, since we just have to use the right rules. As in two-sided systems, $\wedge_R$ and $\vee_R$ are the logical rules and c_R and w_R are the structural rules. The admissible Cut rule has the following shape:

$$\mathrm{Cut} \frac{\vdash \Phi, A \qquad \vdash \Psi, \bar{A}}{\vdash \Phi, \Psi}$$

In GS systems we assume formulas to be in **negation normal form**. A formula is in negation normal form if negation occurs only on atoms. For each formula in classical logic there is an equivalent one in negation normal form because we can use the **De Morgan laws**:

$$\begin{aligned} \neg(p \vee q) &\leftrightarrow (\neg p) \wedge (\neg q) \\ \neg(p \wedge q) &\leftrightarrow (\neg p) \vee (\neg q) \\ \neg(\forall x P(x)) &\leftrightarrow \exists x(\neg P(x)) \\ \neg(\exists x P(x)) &\leftrightarrow \forall x(\neg P(x)) \end{aligned}$$

Using the direction from left to right we can push negation to the atoms. These equalities are valid because of the involutive negation of classical logic. Therefore, we will assume formulas to contain negation only on atoms. We will denote this negation with $\bar{A}$ to distinguish from the general $\neg A$.

In the following I will present the calculus of structures and the system SKSgq. Its soundness and completeness with respect to classical logic are proved by translation to GS1. Cut admissibility comes as a corollary.

2.1 System SKSgq

The framework of the calculus of structures is the language KSq. Formulas of KSq are generated by the following grammar:

$$A ::= f \mid t \mid \alpha \mid [A, A] \mid (A, A) \mid \bar{A} \mid \exists x A \mid \forall x A$$

where f and t are the units **false** and **true**, $[A, B]$ is a disjunction and (A, B) is a conjunction. $\bar{A}$ is the negation of the formula A. In first-order logic there are non-logical symbols too: constants, function symbols and predicate symbols. Let us first define **terms**:

- Variables and constants are terms.
- If $t_1, \cdots, t_n$ are terms and f a n-ary function symbol then $f(t_1, \cdots, t_n)$ is a term.

Terms are interpreted as elements of the underlying field. Predicate symbols denote relations among those elements. The letter α stands for an **atom**, which is a predicate symbol applied to some terms, possibly negated. Atoms are the smallest formulas. Bigger formulas are built by atoms using the logical connectives.

A formula **context**, denoted by $S\{\ \}$, is a formula with one occurrence of $\{\ \}$, a hole, that does not appear in the scope of a negation. For example, $S\{\ \} = (\bar{A}, [\{\ \}, B])$. Consider $\{\ \}$ as a placeholder. $S\{R\}$ denotes the formula obtained by filling the hole in $S\{\ \}$ with R. For the same example, $S\{R\} = (\bar{A}, [R, B])$. We drop the curly braces when they are redundant: for example, $S[R, T]$ is short for $S\{[R, T]\}$. We call R a **subformula** of a formula T if there is a context $S\{\ \}$ such that $S\{R\}$ is T. A deep inference rule ρ is of the shape:

$$\rho \frac{S\{R\}}{S\{T\}}$$

Seen from top to bottom, or from premise to conclusion, it says that whenever the subformula R occurs inside a formula, it can be replaced by the formula T. Thus, in deep inference the active formulas can be arbitrarily deep inside the context. The sequent calculus allows only contexts of the form $\vdash \Phi, \{\ \}$ where Φ is a multiset of formulas. If neither the premise nor the conclusion are inside a context, then the inference rule is called **shallow**.

We know what an inference rule in CoS looks like. Let us now introduce the notion of duality:

Definition 2.1.1. The **dual** of an inference rule is obtained by exchanging premise and conclusion and replacing each connective by its De Morgan dual:

$$\rho\downarrow\frac{S\{R\}}{S\{T\}} \quad \text{is dual to} \quad \rho\uparrow\frac{S\{\bar{T}\}}{S\{\bar{R}\}}$$

A system of inference rules is called **symmetric** if for each of its rules it also contains the dual rule.

In Figure 2.2 we can see the system SKSgq. The first S stands for "symmetric", which means that for every rule we also have its dual. The K stands for "klassisch", as in Gentzen's LK, which indicates that we refer to classical logic. The second S says that it is a system in the calculus of structures. The g is for "general", as opposed to atomic. By this we mean that rules can be applied to arbitrary formulas, not only to atoms. More about atomicity in the next chapter. The q denotes first-order quantifiers. Rules with a name that contains an arrow pointing downward are called **down-rules** and those with the arrow pointing upwards are the **up-rules**. The up-rules are the duals of the down-rules and, except for i↑, they carry the same name prefixed with a "co-".

Rules i↓ and i↑ are the **identity** and the **cut** rule. The identity introduces a pair of dual formulas, as the corresponding rule Ax in the sequent calculus. Of course, i↓ can appear anywhere in a proof, not only at the top. The cut rule eliminates a dual pair, as the Cut rule of GS1. The Axiom and the Cut rule of the sequent calculus are obviously related. The first, when seen top-down, introduces an arbitrary formula A together with its negation $\bar{A}$. The other does the same thing, but this time when seen bottom-up. Furthermore, during Gentzen's cut elimination, cuts are pushed to the top of the proof where they interact with the axioms and vanish. However, their duality is obscured by the top-down asymmetry of the sequent calculus: the Axiom rule has just one premise but the Cut rule is a two-premise inference rule. In deep inference we restore the symmetry so the duality between the two rules is precise: i↑ and ↓ are dual, which means that one can be obtained from the other by exchanging premise and conclusion and negating them.

The structural rules of **weakening** and **contraction** are represented by w↓ and c↓. They are straightforward translations of the structural rules in sequent systems. Co-weakening w↑ and co-contraction c↑ are their duals.

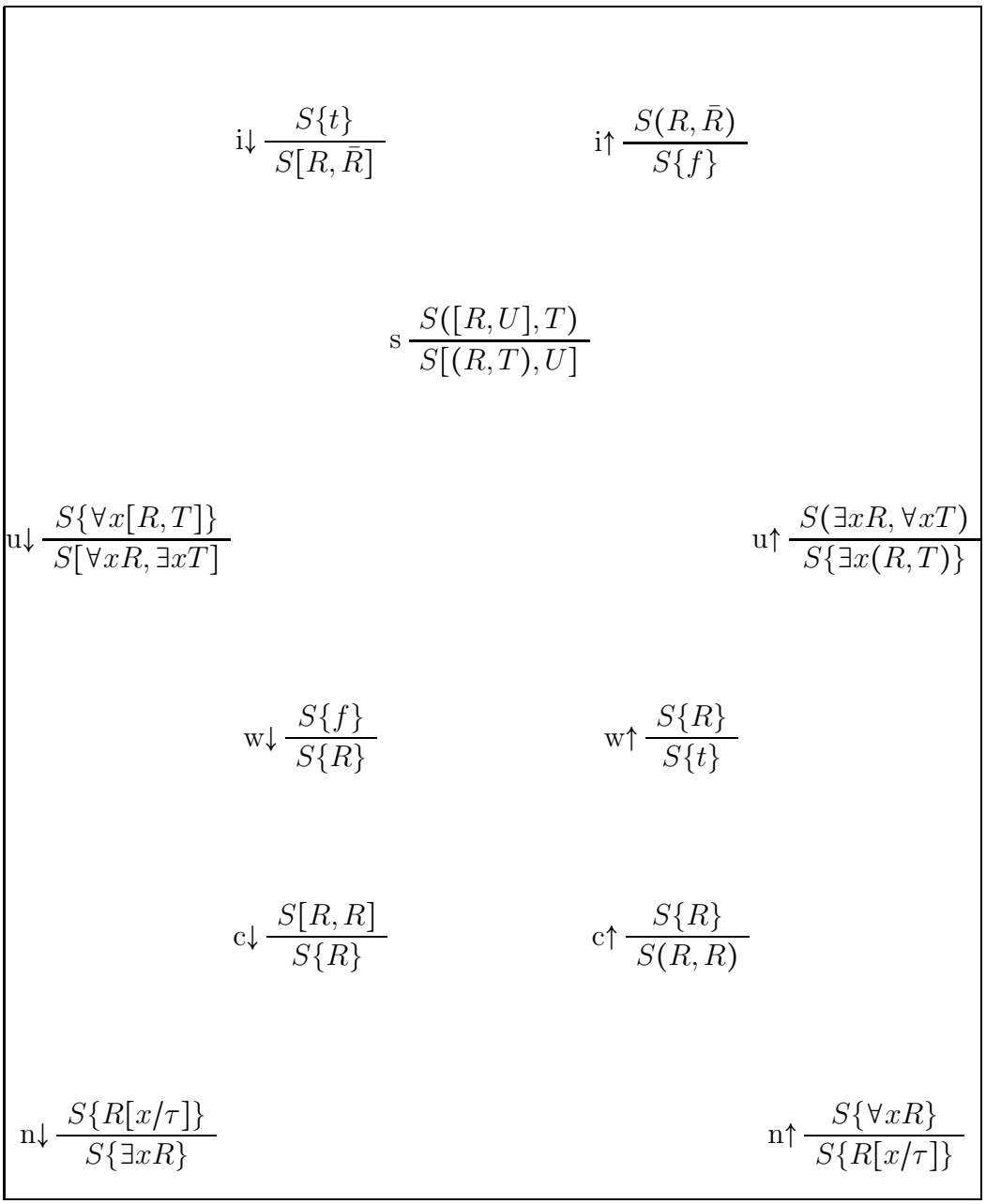


Figure 2.2: System SKSgq

The rule s is called **switch**. It corresponds to the conjunction rule:

$$\wedge_R \frac{\vdash \Phi, A \qquad \vdash \Psi, B}{\vdash \Phi, \Psi, A \wedge B}$$

To see this just consider one of the contexts Φ and Ψ to be empty. Check that the dual of the switch rule is the switch rule itself. It is self-dual.

Rules u$\downarrow$ and n$\downarrow$ are called **universal** and **instantiation** and correspond to the quantifying rules $\forall_R$ and $\exists_R$ respectively. The $\forall_R$ rule of system GS1:

$$\forall_R \frac{\vdash \Phi, A[x/y]}{\vdash \Phi, \forall x A} \quad y \notin FV\{\Phi, \forall x A\}$$

if seen bottom-up, removes the universal quantifier from a formula to allow other rules to access this formula. This is why we have to introduce the **proper** variable y. A proper variable implies a universal quantifier. In system SKSgq inference rules apply deep inside formulas, so there is no need to remove the quantifier. It suffices to be moved out of the way using the rule u$\downarrow$, which does not carry any proviso for any proper variable. In addition, the premise of the u$\downarrow$ rule implies its conclusion, which is not true for the $\forall_R$ rule of the sequent calculus.

In the $\exists_R$ rule of GS1:

$$\exists_R \frac{\vdash \Phi, A[x/\tau]}{\vdash \Phi, \exists x A}$$

the substitution operation requires quantifiers in A not to capture variables in τ. In n$\downarrow$, free variables of the term τ should not be captured by quantifiers in R. However, quantifiers in $S\{\ \}$ may capture variables in τ.

Formulas are considered **syntactically equivalent** modulo the smallest equivalence relation induced by the equations in Figure 2.3. A **structure** is an equivalence class of formulas. In general, we do not distinguish between equivalent formulas so we are actually handling structures, not formulas. But inside a deduction we prefer to think of formulas, so we use an explicit **equivalence rule**:

$$= \frac{T}{R}$$

Equivalence rules do not alter the sense of a proof. They only blow polynomially the size of proofs, without destroying any good property. We will omit obvious instances of the equivalence rule from derivations. We will insist mostly on instances of unit equivalence, variable renaming and vacuous quantifier. The negation equivalences are the symmetry of negation and the laws of De Morgan. Therefore, we can assume formulas to be in negation normal form. For example, instead of the formula $(\overline{[Q,R]},T)$ we use the equivalent $((\bar{Q},\bar{R}),T)$. Thanks to associativity, this can be written $(\bar{Q},\bar{R},T)$.

If we want to restrict our system to propositional logic we should omit the rules u↓, u↑, n↓ and n↑ and obtain the system SKSg. In that case of course atoms are not possibly negated atomic types but possibly negated propositional variables, which are usually called **literals**. The equivalence relation for SKSg will not contain the variable renaming equation, the vacuous quantifier and the instances of negation that refer to predicate logic.

A **derivation** in a system S is a finite sequence of instances of inference rules in the system:

$$\begin{array}{c} \pi\,\dfrac{T}{} \\ \pi'\,\dfrac{V}{} \\ \rho'\,\dfrac{\vdots}{} \\ \rho\,\dfrac{U}{} \\ \hline R \end{array}$$

It is a deduction from T to R. The topmost structure is called the **premise** and the structure at the bottom is called its **conclusion**. We denote a derivation by $\begin{array}{c} T \\ \|\mathrm{s} \\ R \end{array}$. Note that the notion of derivation here is top-down symmetric, contrary to the corresponding notion in the sequent calculus. A **proof** is a derivation whose premise is the unit t. Deep inference allows putting derivations into a context in order to obtain a new derivation. Given a derivation Δ, the derivation $S\{\Delta\}$ is obtained as follows:

- **Associativity**

$$[[R,T],U] = [R,[T,U]]$$
$$((R,T),U) = (R,(T,U))$$

- **Commutativity**

$$[R,T] = [T,R]$$
$$(R,T) = (T,R)$$

- **Units**

$$(f,f) = f \quad [f,R] = R$$
$$[t,t] = t \quad (t,R) = R$$

- **Context**

$$\text{if } R = T \text{ then } S\{R\} = S\{T\}$$
$$\text{and } \bar{R} = \bar{T}$$

- **Negation**

$$\bar{f} = t$$
$$\bar{t} = f$$
$$\overline{[R,T]} = (\bar{R},\bar{T})$$
$$\overline{(R,T)} = [\bar{R},\bar{T}]$$
$$\bar{\bar{R}} = R$$
$$\overline{\exists x R} = \forall x \bar{R}$$
$$\overline{\forall x R} = \exists x \bar{R}$$

- **Variable Renaming**

$$\forall x R = \forall y R[x/y], \quad \text{if } y \text{ is not free in } R.$$
$$\exists x R = \exists y R[x/y], \quad \text{if } y \text{ is not free in } R.$$

- **Vacuous Quantifier**

$$\forall y R = \exists y R = R, \quad \text{if } y \text{ is not free in } R.$$

Figure 2.3: Syntactic Equivalence

$$\Delta = \quad \pi\,\dfrac{T}{\pi'\,\dfrac{V}{\rho'\,\dfrac{\vdots}{\rho\,\dfrac{U}{R}}}} \qquad \rightsquigarrow \qquad S\{\Delta\} = \quad \pi\,\dfrac{S\{T\}}{\pi'\,\dfrac{S\{V\}}{\rho'\,\dfrac{\vdots}{\rho\,\dfrac{S\{U\}}{S\{R\}}}}}$$

This corresponds to adding formulas to the context of every rule instance in a sequent calculus derivation. For example, if we have an instance of contraction:

$$\mathrm{c}_R\,\frac{\vdash \Gamma, A \wedge B, A \wedge B}{\vdash \Gamma, A \wedge B}$$

with context Γ we can obtain another instance by adding the formula C to the context:

$$\mathrm{c}_R\,\frac{\vdash \Gamma, C, A \wedge B, A \wedge B}{\vdash \Gamma, C, A \wedge B}$$

The difference is of course that in the calculus of structures there are no constraints on the context that we will put a derivation into.

The dualisation of inference rules extends to derivations:

Definition 2.1.2. The **dual** of a derivation is obtained by turning it upside-down and replacing each rule, each connective and each atom by its dual.

For example:

$$\mathrm{w}\!\uparrow\,\dfrac{[(\alpha, \bar{b}), \alpha]}{\mathrm{c}\!\downarrow\,\dfrac{[\alpha, \alpha]}{\alpha}} \qquad \text{is dual to} \qquad \mathrm{c}\!\uparrow\,\dfrac{\bar{\alpha}}{\mathrm{w}\!\downarrow\,\dfrac{(\bar{\alpha}, \bar{\alpha})}{([\bar{\alpha}, b], \bar{\alpha})}}$$

The dual of a proof is not a proof. It is a derivation whose conclusion is the unit f. Such a derivation is called **refutation**. Dualising a derivation from T to R, yields a derivation from $\bar{R}$ to $\bar{T}$. This duality is known as **contraposition**. In classical logic the two implications $T \to R$ and $\bar{R} \to \bar{T}$ are equivalent, because they both correspond to the disjunction $\bar{T} \vee R$. Therefore, one could say that the symmetry of CoS extends the symmetry of classical logic from formulas to inference rules and derivations.

Sometimes we use explicitly rules although they can be expressed inside the system:

Definition 2.1.3. A rule ρ is **derivable** for a system S if for every instance of $\rho\,\dfrac{T}{R}$ there is a derivation $\begin{array}{l}T\\ \Vert\,\mathrm{S}\\ R\end{array}$.

We now see that one can easily move back and forth between a derivation and a proof via the following theorem:

Theorem 2.1.1 (**Deduction Theorem**). *There is a derivation* $\begin{array}{l}T\\ \Vert\,\mathit{SKSgq}\\ R\end{array}$ *iff there is a proof* $\begin{array}{l}t\\ \Vert\,\mathit{SKSgq}\\ [\bar{T},R]\end{array}$.

Proof. A proof can be obtained from a given derivation as follows:

$$\begin{array}{l}T\\ \Delta\,\Vert\,\mathrm{SKSgq}\\ R\end{array} \quad\rightsquigarrow\quad \mathrm{i}{\downarrow}\,\dfrac{t}{[\bar{T},T]}\\ \qquad\qquad [\bar{T},\Delta]\,\Vert\,\mathrm{SKSgq}\\ \qquad\qquad [\bar{T},R]$$

If we have the proof, the corresponding derivation is given as follows:

$$\begin{array}{l}t\\ \Pi\,\Vert\,\mathrm{SKSgq}\\ [\bar{T},R]\end{array} \quad\rightsquigarrow\quad \begin{array}{l}=\dfrac{T}{(T,t)}\\ \qquad (\bar{T},\Pi)\,\Vert\,\mathrm{SKSgq}\\ \mathrm{s}\,\dfrac{(T,[\bar{T},R])}{[R,(\bar{T},T)]}\\ \mathrm{i}{\uparrow}\,\dfrac{}{=\dfrac{[R,f]}{R}}\end{array}$$

□

The same theorem holds for the propositional part of GS1:

Theorem 2.1.2. *In system GS1p + Cut there is a derivation* $\begin{array}{l}\vdash T\\ \bigtriangledown\ GS1p{+}Cut\\ \vdash R\end{array}$ *if and only if there is a proof* $\begin{array}{l}\bigtriangledown\ GS1p{+}Cut\\ \vdash \bar{T}, R\end{array}$.

Proof. From left to right, it will be enough to put the given derivation into the context $\vdash \bar{T}, \{\ \}$:

$$\begin{array}{l}\text{Ax}\ \dfrac{}{\vdash \bar{T}, T}\\ \bigtriangledown\ \text{GS1p+Cut}\\ \vdash \bar{T}, R\end{array}$$

This is a proof of $\vdash \bar{T}, R$.
From right to left, we can build the following derivation:

$$\begin{array}{l}\bigtriangledown\ \text{GS1p+Cut}\\ \text{Cut}\ \dfrac{\vdash \bar{T}, R \qquad\qquad \vdash T}{\vdash R}\end{array}$$

□

The proof is completely analogue to that for CoS: from left to right we add the negated premise throughout the proof, and from right to left we just use a cut. However, this method does not work for system GS1. The direction from left to right fails because of the $\forall_R$ rule: adding formulas to the context of a derivation can violate its proviso. In SKSgq there was no problem because the provisos for the equations of variable renaming and vacuous quantifier only require checking the subformula that is being changed, while the proviso of the $\forall_R$ rule requires checking the entire context.

The proof of the deduction theorem for CoS is also reminiscent of the proof of the same theorem for the **Hilbert proof systems**. This is historically the first method for formal reasoning, attributed to Gottlob Frege and David Hilbert. As a formalism, it is characterised by the choice of a large number of logical axiom schemes and a small set of rules of inference. We can see a Hilbert system for classical propositional logic in Figure 2.1. There are three axiom schemes and the only inference rule is **modus ponens**. Note that the

only connectives that appear are $\rightarrow$ and $\neg$. We can do this because $\{\rightarrow, \neg\}$ is an **adequate** set of connectives, that is every logical connective can be expressed in terms of implication and negation:

$$A \vee B \leftrightarrow \neg A \rightarrow B$$
$$A \wedge B \leftrightarrow \neg(A \rightarrow \neg B)$$

These equivalences are of course due to the symmetry of classical logic.

A formal proof is a finite sequence of formulas in which each formula is either an axiom or is obtained from previous formulas by MP. For example, this is a proof of the (quite obvious) sentence $\varphi \rightarrow \varphi$:

$1.[\varphi \rightarrow ((\varphi \rightarrow \varphi) \rightarrow \varphi)] \rightarrow [(\varphi \rightarrow (\varphi \rightarrow \varphi)) \rightarrow (\varphi \rightarrow \varphi)]$	Ax2
$2.\ \varphi \rightarrow ((\varphi \rightarrow \varphi) \rightarrow \varphi)$	Ax1
$3.(\varphi \rightarrow (\varphi \rightarrow \varphi)) \rightarrow (\varphi \rightarrow \varphi)$	MP:1, 2
$4.(\varphi \rightarrow (\varphi \rightarrow \varphi))$	Ax1
$5.\ \varphi \rightarrow \varphi$	MP:3, 4

The existence of this proof is denoted as $\vdash_{Hc} \varphi \rightarrow \varphi$. If in a proof of a sentence ψ we use a set T of non-logical axioms, axioms that do not occur from the axiom schemes of Hc, we write $T \vdash_{Hc} \psi$. Classically valid are those sentences that can be proved using only the logical axioms of the system.

Ax1: $\varphi \rightarrow (\psi \rightarrow \varphi)$

Ax2: $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$

Ax3: $(\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi)$

MP $\dfrac{\varphi \rightarrow \psi \qquad \varphi}{\psi}$

Figure 2.4: System Hc

Let us see now the deduction theorem for the Hilbert formalism:

Theorem 2.1.3. *If* $T \cup \{\varphi\} \vdash_{Hc} \psi$ *then* $T \vdash_{Hc} \varphi \to \psi$.

Proof. Consider a set T of non-logical axioms and suppose that $T \cup \{\varphi\} \vdash_{Hc} \psi$. Thus there is a formal proof $\psi_1, \cdots, \psi_n\,(= \psi)$ with axioms $T \cup \{\varphi\}$. We will prove by induction on the length of the given proof that for all $i = 1, \cdots, n$: $T \vdash_{Hc} \varphi \to \psi_i$.

- For $i = 1$: If ψ_1 is a logical or non-logical axiom then $T \vdash_{Hc} \varphi \to \psi_1$ as seen in the following proof:

  $$\begin{array}{ll} 1.\psi_1 \to (\varphi \to \psi_1) & \text{Ax1} \\ 2.\psi_1 & \text{Ax or } T \\ 3.\varphi \to \psi_1 & \text{MP:}1, 2 \end{array}$$

  If $\psi_1 = \varphi$ then $\vdash_{Hc} \varphi \to \psi_1$ because $\vdash_{Hc} \varphi \to \varphi$ (see the example proof). So we can write $T \vdash_{Hc} \varphi \to \psi_1$.

- Now suppose that for every $k < i$ we have $T \vdash_{Hc} \varphi \to \psi_k$ (induction hypothesis). If ψ_i is an axiom or $\psi_i = \varphi$ then we prove $T \vdash_{Hc} \varphi \to \psi_i$ as before. If ψ_i is the conclusion of an MP instance using the premisses $\psi_j, \psi_l = \psi_j \to \psi_i\,(j, l < i)$ then, by induction hypothesis, we know that $T \vdash_{Hc} \varphi \to \psi_j$ and $T \vdash_{Hc} \varphi \to (\psi_j \to \psi_i)$. So we can built a proof of $\varphi \to \psi_i$ as follows:

  $$\begin{array}{ll} 1. & \cdots \\ \cdots & \cdots \\ m. & \varphi \to \psi_j \\ \cdots & \cdots \\ \cdots & \cdots \\ n. & \varphi \to (\psi_j \to \psi_i) \\ n+1.(\varphi \to (\psi_j \to \psi_i)) \to ((\varphi \to \psi_j) \to (\varphi \to \psi_i)) & \text{Ax2} \\ n+2.(\varphi \to \psi_j) \to (\varphi \to \psi_i) & \text{MP:}n, n+1 \\ n+3.\ \varphi \to \psi_i & \text{MP:}m, n+2 \end{array}$$

  Thus $T \vdash_{Hc} \varphi \to \psi_i$.

We proved inductively that for all $i = 1, \cdots, n$: $T \vdash_{Hc} \varphi \to \psi_i$. Therefore, $T \vdash_{Hc} \varphi \to \psi$ and the proof is complete. □

What we did above is similar to what happens in CoS. We took the formal proof:

$$\begin{array}{c} \psi_1 \\ \vdots \\ \psi_n \end{array}$$

from axioms $T \cup \{\varphi\}$ and showed the existence of the proofs:

$$\begin{array}{c} T \vdash_{Hc} \varphi \to \psi_1 \\ \vdots \\ T \vdash_{Hc} \varphi \to \psi_n \end{array}$$

This could be seen as putting the given proof into the context $\varphi \to \{\ \}$ and dropping φ from the hypotheses. The same as we did in the case of SKSgq, where we put a given derivation $\begin{array}{c} Q \\ \| \\ R \end{array}_{\text{SKSgq}}$ into the context $[\bar{Q}, \{\ \}]$ and obtained a proof of $[\bar{Q}, R]$. Of course, $[\bar{Q}, \{\ \}]$ is semantically equivalent to $Q \to \{\ \}$. This theorem can be proved the same way for the predicate calculus.

2.2 Correspondence to the Sequent Calculus

We will translate derivations of GS1 + Cut to derivations in SKSgq and vice versa. But the system GS1 is equivalent to G1c and thus sound and complete with respect to classical logic. This will mean that system SKSgq is sound and complete for classical predicate logic as well.

Soundness

When we translate between different formal languages we use a translation mapping:

Definition 2.2.1. The function $\underline{\cdot}_G$ maps formulas and sequents of KSq to formulas of GS1:

$$\begin{aligned}
\underline{\alpha}_G &= \alpha \\
\underline{t}_G &= \top \\
\underline{f}_G &= \bot \\
\underline{[R,T]}_G &= \underline{R}_G \vee \underline{T}_G \\
\underline{(R,T)}_G &= \underline{R}_G \wedge \underline{T}_G \\
\underline{\exists x A}_G &= \exists x \underline{A}_G \\
\underline{\forall x A}_G &= \forall x \underline{A}_G
\end{aligned}$$

Before the soundness theorem we need to prove the following lemma:

Lemma 2.2.1. *For every two formulas A, B and every formula context $C\{\ \}$ there is a derivation:*

$$\vdash A, \bar{B}$$

$$\bigtriangledown$$

$$\vdash C\{A\}, \overline{C\{B\}}$$

in GS1.

Proof. By structural induction on the context $C\{\ \}$:

- Base case: $C\{\ \} = \{\ \}$. Sometimes we call that the **empty context**. Of course there is such a derivation since the premise and the conclusion of the derivation are the same.

- Now suppose that $C\{\ \} = C_1 \wedge C_2\{\ \}$, and the induction hypothesis holds for $C_2\{\ \}$. Then we have:

$$\dfrac{\dfrac{\dfrac{}{\vdash C_1, \overline{C_1}}\,\mathrm{Ax} \qquad \dfrac{\vdash A, \bar{B}}{\bigtriangledown}\,\Delta \vdash C_2\{A\}, \overline{C_2\{B\}}}{\vdash C_1 \wedge C_2\{A\}, \overline{C_1}, \overline{C_2\{B\}}}\,\wedge_R}{\vdash C_1 \wedge C_2\{A\}, \overline{C_1} \vee \overline{C_2\{B\}}}\,\vee_R$$

  where Δ exists by induction hypothesis.

- If $C\{\ \} = C_1 \vee C_2\{\ \}$:

$$\dfrac{\dfrac{\dfrac{}{\vdash C_1, \overline{C_1}}\,\mathrm{Ax} \qquad \dfrac{\vdash A, \bar{B}}{\bigtriangledown}\,\Delta \vdash C_2\{A\}, \overline{C_2\{B\}}}{\vdash C_1, C_2\{A\}, \overline{C_1} \wedge \overline{C_2\{B\}}}\,\wedge_R}{\vdash C_1 \vee C_2\{A\}, \overline{C_1} \wedge \overline{C_2\{B\}}}\,\vee_R$$

  where Δ exists by induction hypothesis.

- If $C\{\ \} = \exists x C'\{\ \}$:

$$\dfrac{\dfrac{\dfrac{\vdash A, \bar{B}}{\bigtriangledown}\,\Delta \vdash C'\{A\}, \overline{C'\{B\}}}{\vdash \exists x C'\{A\}, \overline{C'\{B\}}}\,\exists_R}{\vdash \exists x C'\{A\}, \forall x \overline{C'\{B\}}}\,\forall_R$$

  where Δ exists by induction hypothesis.

- If $C\{\ \} = \forall x C'\{\ \}$:

$$\dfrac{\dfrac{\vdash A, \bar{B}}{\vdash C'\{A\}, \overline{C'\{B\}}}\,\Delta}{\dfrac{\vdash C'\{A\}, \exists x \overline{C'\{B\}}}{\vdash \forall x C'\{A\}, \exists x \overline{C'\{B\}}}\,\forall_R}\,\exists_R$$

  where Δ exists by induction hypothesis.

□

Now we will see how we can imitate deep inference in the sequent calculus:

Theorem 2.2.2 (**Soundness**)**.** *For every derivation in SKSgq there exists a corresponding derivation in GS1 + Cut:*

$$\begin{array}{c} Q \\ \Delta \,\|\, SKSgq \\ P \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \vdash \underline{Q}_G \\ \Big\triangledown\, GS1{+}Cut \\ \vdash \underline{P}_G \end{array}$$

Proof. We will prove the theorem by induction on the length of the given derivation Δ in SKSgq. In proofs we will usually drop the subscript of the translation to improve readability.

- If the derivation Δ consists of just one formula P then the corresponding derivation in GS1 consists of just one sequent $\vdash P$.
- We single out the topmost rule instance in Δ:

$$\begin{array}{c} Q \\ \Delta \,\|\, \mathrm{SKSgq} \\ P \end{array} \quad = \quad \dfrac{S\{T\}}{S\{R\}}\,\rho \qquad \begin{array}{c} \\ \Delta' \,\|\, \mathrm{SKSgq} \\ P \end{array}$$

The corresponding derivation in GS1 will be as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash R, \bar{T}}\,\Pi}{}\qquad \dfrac{\dfrac{}{\vdash S\{R\}, \overline{S\{T\}}}\,\Delta_1 \qquad \vdash S\{T\}}{\vdash S\{R\}}\,\mathrm{Cut}}{}\qquad \dfrac{}{\vdash P}\,\Delta_2}{}$$

where Δ_1 exists by Lemma 2.2.1 and Δ_2 exists by induction hypothesis. The proof Π depends on the rule ρ. Thus it will be enough to show that for every rule $\rho\,\dfrac{S\{T\}}{S\{R\}}$ of SKSgq there exists a proof $\dfrac{}{\vdash R, \bar{T}}\,\Pi$ in GS1:

$$\mathrm{i}{\downarrow}\,\frac{S\{t\}}{S[R, \bar{R}]} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{}{\vdash R, \bar{R}}\,\mathrm{Ax}}{\vdash R \vee \bar{R}}\,\vee_R}{\vdash R \vee \bar{R}, \bot}\,\mathrm{w}_R$$

$$\mathrm{i}{\uparrow}\,\frac{S(R, \bar{R})}{S\{f\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{}{\vdash R, \bar{R}}\,\mathrm{Ax}}{\vdash R \vee \bar{R}}\,\vee_R}{\vdash R \vee \bar{R}, \bot}\,\mathrm{w}_R$$

$$\mathrm{s}\,\frac{S([R, U], T)}{S[(R, T), U]} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash R, \bar{R}}\,\mathrm{Ax} \qquad \dfrac{}{\vdash U, \bar{U}}\,\mathrm{Ax}}{\vdash R, U, \bar{R} \wedge \bar{U}}\,\wedge_R \qquad \dfrac{}{\vdash T, \bar{T}}\,\mathrm{Ax}}{\vdash R \wedge T, \bar{R} \wedge \bar{U}, U, \bar{T}}\,\wedge_R}{\vdash (R \wedge T) \vee U, \bar{R} \wedge \bar{U}, \bar{T}}\,\vee_R}{\vdash (R \wedge T) \vee U, (\bar{R} \wedge \bar{U}) \vee \bar{T}}\,\vee_R$$

$$\mathrm{w}{\downarrow}\,\frac{S\{f\}}{S\{R\}} \qquad \rightsquigarrow \qquad \mathrm{w}_R\,\frac{\top\,\dfrac{}{\vdash \top}}{\vdash R, \top}$$

$$\mathrm{w}{\uparrow}\,\frac{S\{R\}}{S\{t\}} \qquad \rightsquigarrow \qquad \mathrm{w}_R\,\frac{\top\,\dfrac{}{\vdash \top}}{\vdash \bar{R}, \top}$$

$$\mathrm{c}{\downarrow}\,\frac{S[R,R]}{S\{R\}} \qquad \rightsquigarrow \qquad \dfrac{\wedge_R\,\dfrac{\mathrm{Ax}\,\dfrac{}{\vdash R, \bar{R}} \qquad \mathrm{Ax}\,\dfrac{}{\vdash R, \bar{R}}}{\mathrm{c}_R\,\dfrac{\vdash R, R, \bar{R} \wedge \bar{R}}{\vdash R, \bar{R} \wedge \bar{R}}}}{}$$

$$\mathrm{c}{\uparrow}\,\frac{S\{R\}}{S(R,R)} \qquad \rightsquigarrow \qquad \dfrac{\wedge_R\,\dfrac{\mathrm{Ax}\,\dfrac{}{\vdash R, \bar{R}} \qquad \mathrm{Ax}\,\dfrac{}{\vdash R, \bar{R}}}{\mathrm{c}_R\,\dfrac{\vdash \bar{R}, \bar{R}, R \wedge R}{\vdash \bar{R}, R \wedge R}}}{}$$

$$\mathrm{u}{\downarrow}\,\frac{S\{\forall x[R,T]\}}{S[\forall x R, \exists x T]} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\wedge\,\dfrac{\mathrm{Ax}\,\dfrac{}{\vdash R, \bar{R}} \qquad \mathrm{Ax}\,\dfrac{}{\vdash T, \bar{T}}}{\vdash R, T, \bar{R} \wedge \bar{T}}}{\exists_R\,\dfrac{}{\vdash R, \exists x T, \bar{R} \wedge \bar{T}}}}{\exists_R\,\dfrac{}{\vdash R, \exists x T, \exists x(\bar{R} \wedge \bar{T})}}}{\forall_R\,\dfrac{}{\vdash \forall x R, \exists x T, \exists x(\bar{R} \wedge \bar{T})}}}{\vee_R\,\dfrac{}{\vdash \forall x R \vee \exists x T, \exists x(\bar{R} \wedge \bar{T})}}$$

$$\mathrm{u}\uparrow \frac{S(\exists xR, \forall xT)}{S\{\exists x(R,T)\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash R, \bar{R}}\,\mathrm{Ax} \quad \dfrac{}{\vdash T, \bar{T}}\,\mathrm{Ax}}{\vdash R \wedge T, \bar{R}, \bar{T}}\,\wedge}{\vdash R \wedge T, \bar{R}, \exists x\bar{T}}\,\exists_R}{\vdash \exists x(R \wedge T), \bar{R}, \exists x\bar{T}}\,\exists_R}{\vdash \exists x(R \wedge T), \forall x\bar{R}, \exists x\bar{T}}\,\forall_R}{\vdash \exists x(R \wedge T), \forall x\bar{R} \vee \exists x\bar{T}}\,\vee_R$$

$$\mathrm{n}\downarrow \frac{S\{R[x/\tau]\}}{S\{\exists xR\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{}{\vdash R[x/\tau], \overline{R[x/\tau]}}\,\mathrm{Ax}}{\vdash \exists xR, \overline{R[x/\tau]}}\,\exists_R$$

$$\mathrm{n}\uparrow \frac{S\{\forall xR\}}{S\{R[x/\tau]\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{}{\vdash R[x/\tau], \overline{R[x/\tau]}}\,\mathrm{Ax}}{\vdash R[x/\tau], \exists x\overline{R[x/\tau]}}\,\exists_R$$

□

Notice that the simulation of deep inference in the sequent calculus is done by using the Cut rule. Also observe that for each dual pair of rules the two sequent proofs are identical. This is not surprising. If a rule of CoS describes an implication $A \to B$ then its dual rule gives the implication $\bar{B} \to \bar{A}$, which is equivalent in classical logic. Both implications correspond to the one-sided sequent $\vdash B, \bar{A}$. Therefore, the translations of the up-rules occur from the translations of the down-rules but with all atoms negated.

Another remark is that in the induction on the length of a given derivation in SKSgq we took cases according to the topmost rule and not the last one, as usual. We can do that because derivations in CoS are top-down symmetric. In other words, there is only one premise. So we are able to isolate it and apply the induction hypothesis on the derivation below it.

We couldn't have done that in the sequent calculus, where each derivation has many leaves (premisses). Another way to think of this is the following: applying an induction considering the topmost rule is equivalent to applying an induction on the dual derivation considering the bottom rule.

The following result comes very easily since proofs are special derivations:

Corollary 2.2.3. *If a formula S has a proof in SKSgq then $\underline{S}_G$ has a proof in GS1.*

Therefore, SKSgq is sound with respect to classical logic.

Completeness

For this, we need to translate expressions of GS1 to their equivalent in the language of CoS:

Definition 2.2.2. The function $\underline{\cdot}_S$ maps formulas and sequents of GS1 to formulas of KSq:

$$\begin{aligned}
\underline{\alpha}_S &= \alpha \\
\underline{\top}_S &= t \\
\underline{\bot}_S &= f \\
\underline{A \vee B}_S &= [\underline{A}_S, \underline{B}_S] \\
\underline{A \wedge B}_S &= (\underline{A}_S, \underline{B}_S) \\
\underline{\exists x A}_S &= \exists x \underline{A}_S \\
\underline{\forall x A}_S &= \forall x \underline{A}_S \\
\underline{A_1, ..., A_n}_S &= [\underline{A_1}_S, ..., \underline{A_n}_S] \\
\underline{\varnothing}_S &= f
\end{aligned}$$

Theorem 2.2.4 (**Completeness**)**.** *For every derivation in GS1 + Cut there exists a derivation in SKSgq\{c↑,w↑,u↑,n↑} with the same number of cuts:*

$$\begin{array}{c} \vdash \Sigma_1 \ \cdots \vdash \Sigma_k \\ \bigtriangledown \ GS1{+}Cut \\ \vdash \Sigma \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \forall x_1 \cdots \forall x_n (\underline{\Sigma_1}_S, \cdots, \underline{\Sigma_k}_S) \\ \Big\| \ SKSgq \setminus \{w\uparrow, c\uparrow, u\uparrow, n\uparrow\} \\ \underline{\Sigma}_S \end{array}$$

where $x_1, ..., x_n$ are the free variables in the premisses $\Sigma_1, ..., \Sigma_k$ that have been introduced by $\forall_R$ instances.

Proof. By structural induction on the given derivation Δ. As we did with the $\underline{\cdot}_G$ translation, we will omit the subscript for simplicity.

- The base cases are the following:

$$\vdash \Sigma \qquad \rightsquigarrow \qquad \Sigma$$

$$\top \frac{}{\vdash \top} \qquad \rightsquigarrow \qquad t$$

$$\mathrm{Ax} \frac{}{\vdash A, \bar{A}} \qquad \rightsquigarrow \qquad \mathrm{i}{\downarrow} \frac{t}{[A, \bar{A}]}$$

- Now we will take cases depending on which rule is the last of the derivation. The derivation above the last rule will be considered translated to SKSgq (induction hypothesis):

$$\begin{array}{c} \vdash \Sigma_1 \ \cdots \vdash \Sigma_k \\ \bigtriangledown \\ \mathrm{c}_R \dfrac{\vdash \Phi, A, A}{\vdash \Phi, A} \end{array} \qquad \rightsquigarrow \qquad \begin{array}{c} \forall x_1 \cdots \forall x_n (\Sigma_1, \cdots, \Sigma_k) \\ \Big\| \scriptstyle{\mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}} \\ \mathrm{c}{\downarrow} \dfrac{[\Phi, A, A]}{[\Phi, A]} \end{array}$$

$$\begin{array}{c} \vdash \Sigma_1 \ \cdots \vdash \Sigma_k \\ \bigtriangledown \\ \mathrm{w}_R \dfrac{\vdash \Phi}{\vdash \Phi, A} \end{array} \qquad \rightsquigarrow \qquad \begin{array}{c} \forall x_1 \cdots \forall x_n (\Sigma_1, \cdots, \Sigma_k) \\ \Big\| \scriptstyle{\mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}} \\ = \dfrac{\Phi}{[\Phi, f]} \\ \mathrm{w}{\downarrow} \dfrac{}{[\Phi, A]} \end{array}$$

$$\dfrac{\vdash \Sigma_1 \ \cdots \vdash \Sigma_l \quad \vdash \Sigma_{l+1} \ \cdots \vdash \Sigma_k}{\wedge_R \dfrac{\vdash \Phi, A \qquad \vdash \Psi, B}{\vdash \Phi, \Psi, A \wedge B}}$$

$$\rightsquigarrow$$

$$\dfrac{\forall x_1 \cdots \forall x_n((\Sigma_1, \cdots, \Sigma_l) \ , \quad (\Sigma_{l+1}, \cdots, \Sigma_k))}{\mathrm{s}\dfrac{([\Phi, A] \qquad , \qquad [\Psi, B])}{\mathrm{s}\dfrac{[\Phi, (A, [\Psi, B])]}{[\Phi, \Psi, (A, B)]}}} \ \Big\| \mathrm{SKSgq}\setminus\{w\uparrow,\mathrm{c}\uparrow,\mathrm{u}\uparrow,\mathrm{n}\uparrow\} \ \Big\| \mathrm{SKSgq}\setminus\{w\uparrow,\mathrm{c}\uparrow,\mathrm{u}\uparrow,\mathrm{n}\uparrow\}$$

$$\dfrac{\vdash \Sigma_1 \ \cdots \vdash \Sigma_k}{\vee_R \dfrac{\vdash \Phi, A, B}{\vdash \Phi, A \vee B}} \quad \rightsquigarrow \quad \dfrac{\forall x_1 \cdots \forall x_n(\Sigma_1, \cdots, \Sigma_k)}{= \dfrac{[\Phi, A, B]}{[\Phi, [A, B]]}} \ \Big\| \mathrm{SKSgq}\setminus\{w\uparrow,\mathrm{c}\uparrow,\mathrm{u}\uparrow,\mathrm{n}\uparrow\}$$

For the $\forall_R$ case, we will have to bound the proper variable. Consider the given derivation in GS1 + Cut:

$$\dfrac{\vdash \Sigma_1 \ \cdots \vdash \Sigma_k}{\forall_R \dfrac{\vdash \Phi, A[x/y]}{\vdash \Phi, \forall x A}}$$

By induction hypothesis there is a derivation:

$$\begin{array}{c} \forall x_1 \cdots \forall x_n (\Sigma_1, \cdots, \Sigma_k) \\ \Delta \Big\| \{\mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}\} \\ [\Phi, A[x/y]] \end{array}$$

If we bound y we build the derivation:

$$\begin{array}{c} \forall y \forall x_1 \cdots \forall x_n (\Sigma_1, \cdots, \Sigma_k) \\ \forall y\{\Delta\} \Big\| \{\mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}\} \\ \mathrm{u}\downarrow \dfrac{\forall y [\Phi, A[x/y]]}{= \dfrac{[\exists y \Phi, \forall y A[x/y]]}{= \dfrac{[\Phi, \forall y A[x/y]]}{[\Phi, \forall x A]}}} \end{array}$$

This is what we wanted. In the upper instance of the equivalence rule y is not free in Φ and in the lower instance y is not free in $\forall x A$. We know both because of the proviso of the R$\forall$ rule.

There two more cases remaining:

$$\mathrm{Cut} \dfrac{\begin{array}{cc} \vdash \Sigma_1 \ \cdots \vdash \Sigma_l & \vdash \Sigma_{l+1} \ \cdots \vdash \Sigma_k \\ \bigtriangledown & \bigtriangledown \\ \vdash \Phi, A & \vdash \Psi, \bar{A} \end{array}}{\vdash \Phi, \Psi}$$

$$\wr$$

$$\begin{array}{c} \forall x_1 \cdots \forall x_n ((\Sigma_1, \cdots, \Sigma_l) \ , \quad (\Sigma_{l+1}, \cdots, \Sigma_k)) \\ \Big\| \{\mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}\} \Big\| \{\mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}\} \\ \mathrm{s} \dfrac{([\Phi, A] \quad , \quad [\Psi, \bar{A}])}{\mathrm{s} \dfrac{[\Phi, (A, [\Psi, \bar{A}])]}{\mathrm{i}\uparrow \dfrac{[\Phi, \Psi, (A, \bar{A})]}{= \dfrac{[\Phi, \Psi, f]}{[\Phi, \Psi]}}}} \end{array}$$

$$\begin{array}{c} \vdash \Sigma_1 \ \cdots \vdash \Sigma_k \\ \bigtriangledown \\ \exists_R \dfrac{\vdash \Phi, A[x/\tau]}{\vdash \Phi, \exists x A} \end{array} \quad \rightsquigarrow \quad \begin{array}{c} \forall x_1 \cdots \forall x_n (\Sigma_1, \cdots, \Sigma_k) \\ \Big\| {\scriptstyle \mathrm{SKSgq} \setminus \{w\uparrow, \mathrm{c}\uparrow, \mathrm{u}\uparrow, \mathrm{n}\uparrow\}} \\ \mathrm{n}\downarrow \dfrac{[\Phi, A[x/\tau]]}{[\Phi, \exists x A]} \end{array}$$

Observe that the only rule that requires the i↑ rule in its translation is the Cut rule. Therefore, a cut-free derivation in GS1 will be translated into a cut-free derivation in SKSgq. □

The following corollary establishes completeness:

Corollary 2.2.5. *If a sequent Σ has a proof in GS1 + Cut then $\underline{\Sigma}_S$ has a proof in SKSgq$\setminus\{c\uparrow, w\uparrow, u\uparrow, n\uparrow\}$ with the same number of cuts.*

Thus, SKSgq is complete with respect to classical predicate logic.

2.3 Cut Admissibility

In this section we see that if we care only for proofs, the up-fragment of SKSgq is superfluous. Let us remove all the up-rules from SKSgq. Then we obtain the asymmetric, cut-free system KSgq shown in Figure 2.5. The propositional fragment KSg occurs if we omit u↓ and n↓.

$$\mathrm{i}{\downarrow}\,\frac{S\{t\}}{S[R,\bar{R}]} \qquad \mathrm{w}{\downarrow}\,\frac{S\{f\}}{S\{R\}} \qquad \mathrm{c}{\downarrow}\,\frac{S[R,R]}{S\{R\}}$$

$$\mathrm{s}\,\frac{S([R,U],T)}{S[(R,T),U]} \qquad \mathrm{u}{\downarrow}\,\frac{S\{\forall x[R,T]\}}{S[\forall xR,\exists xT]} \qquad \mathrm{n}{\downarrow}\,\frac{S\{R[x/\tau]\}}{S\{\exists xR\}}$$

Figure 2.5: System KSgq

Definition 2.3.1. A rule ρ is **admissible** for a system S if for every proof $\begin{array}{l} t \\ \Vert_{\mathrm{S}\cup\{\rho\}} \\ A \end{array}$ there is a proof $\begin{array}{l} t \\ \Vert_{\mathrm{S}} \\ A \end{array}$.

It follows that a derivable rule is admissible but an admissible rule is not necessarily derivable.

From Theorem 2.2.4 we already know that c↑, w↑, u↑ and n↑ are admissible for KSgq because we can translate every derivation, and therefore every proof, of GS1 without using them. This is not surprising since the up-rules are the contrapositive versions of the corresponding down-rules. Thus they do not offer any new information. We can show that the cut rule i↑ is also admissible if we rely on the cut elimination for GS1:

Theorem 2.3.1 (**Cut Admissibility**). *The cut rule i↑ is admissible for system KSgq.*

Proof. 1. Consider a given proof $\begin{array}{c} t \\ \| \\ A \end{array}_{\text{SKSgq}}$.

2. By Theorem 2.2.2 we can translate it and obtain a proof of $\vdash A$ in the GS1 + Cut.

3. The Cut rule of sequent systems can be eliminated from any proof. After that procedure we have a proof of $\vdash A$ in GS1.

4. We translate it back to SKSgq. By Theorem 2.2.4 we know that no cuts are added so we obtain a proof without any instances of i↑.

The procedure is shown in Figure 2.6. □

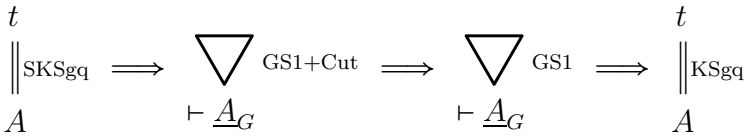


Figure 2.6: Cut Elimination Through Translation

Let us now introduce some notions of equivalence between proof systems:

Definition 2.3.2. Two systems S_1 and S_2 are (weakly) **equivalent** if for every proof $\begin{array}{c} t \\ \| \\ R \end{array}_{S_1}$ there is a proof $\begin{array}{c} t \\ \| \\ R \end{array}_{S_2}$ and vice versa.

We showed above that the up-fragment of SKSgq is admissible. This means that systems SKSgq and KSgq are equivalent.

Definition 2.3.3. Two systems S_1 and S_2 are **strongly equivalent** if for every derivation $\begin{array}{c} Q \\ \| \\ R \end{array}_{S_1}$ there is a derivation $\begin{array}{c} Q \\ \| \\ R \end{array}_{S_2}$ and vice versa.

When a formula R implies a formula T then there is not necessarily a derivation from R to T in KSgq, while there is one in SKSgq. For example, the cut rule i↑ can not be derived in system KSgq. Therefore, systems SKSgq and KSgq are not strongly equivalent.

We have proved that the down-fragment is complete, in the sense that it has a proof for each valid formula. The up-fragment is also complete, because it has a refutation for each unsatisfiable formula. A formula is **unsatisfiable** if it is unprovable under any premise, unless equivalent to f. This is the opposite of validity, if R is unsatisfiable then $\bar{R}$ is valid and vice versa. To see the completeness of the up-fragment, assume that R is unsatisfiable. Then $\bar{R}$ is valid:

$$\begin{array}{c} t \\ \Vert\downarrow \\ \bar{R} \end{array}$$

The dual derivation is the refutation:

$$\begin{array}{c} R \\ \Vert\uparrow \\ f \end{array}$$

So for each unsatisfiable formula there is a refutation in the up-fragment. Now, suppose that we want to prove a valid formula in it. The formula's negation will be of course an unsatisfiable formula and all we have to do is to build its refutation. If we generalise this argument to arbitrary derivations we can conclude that the up-fragment and the down-fragment are strongly equivalent:

$$\begin{array}{c} P \\ \Vert\downarrow \\ Q \end{array} \qquad \leftrightsquigarrow \qquad \begin{array}{c} \bar{Q} \\ \Vert\uparrow \\ \bar{P} \end{array}$$

Every deduction derivable in the one fragment is also derivable in the other in its contrapositive version.

Chapter 3

Locality

Inference rules that deal with an unbounded quantity of information are problematic from the points of view of complexity and implementation. Let's see an example of such a rule in the sequent calculus:

$$\mathrm{c}_R \frac{\vdash \Phi, A, A}{\vdash \Phi, A}$$

Here, going from bottom to top, a formula A of unbounded size is duplicated. Alternatively, if we see the rule top-down, we have to check if two formulas of arbitrary size are identical in order to apply contraction. We will call **local** those inference rules that do not require a global view on formulas of unbounded size, and non-local those rules that do. Rule c_R is obviously a non-local rule.

Another case of non-locality are the context-sharing rules in various sequent systems. Observe the $\wedge_R$ rule of GS1:

$$\wedge_R \frac{\vdash \Phi, A \qquad \vdash \Psi, B}{\vdash \Phi, \Psi, A \wedge B}$$

The contexts of the premisses are independent and in the conclusion they are simply joined together. Rules with such a treatment of contexts are called **context-free** or **multiplicative**. Now consider the following variant of $\wedge_R$:

$$\wedge_R \frac{\vdash \Phi, A \qquad \vdash \Phi, B}{\vdash \Phi, A \wedge B}$$

Here the contexts in both premisses are the same. These rules are called **context-sharing** or **additive**. The two versions are equivalent. Generally, every logical rule in the sequent calculus has two equivalent forms, one

additive and one multiplicative. Their equivalence is shown using the structural rules of weakening and contraction. Let us mention that in linear logic, where these rules are not sound, we have two discrete versions for each classical connective, one additive and one multiplicative. It is clear that additive rules are non-local because, going bottom-up, a context of unbounded size has to be duplicated.

There are two reasons why such a global behaviour is undesirable. First, say that we want to measure the computational effort required for proof-checking. The effort required for checking the correctness of a given instance of the contraction rule depends on the size of the formula that is duplicated. Thus, the usual measures on proofs, like the depth or the number of instances of inference rules, are not suitable for the complexity of proof-checking. A good measure would be more complicated, as it would have to look inside the rule instances. Locality implies a bounded computational cost of applying an inference rule. Second, say that we want to implement contraction on a distributed system, where each processor has a limited amount of local memory. The formula A could be spread over a number of processors. In that case, no single processor has a global view on it. Given a suitable implementation, both of these objections become irrelevant. It is possible to represent sequents in such a way that the contraction rule can be proof-checked in constant time just as it is possible to let several processors duplicate a formula which is distributed among them. However, all the problems of a proof-theoretic system that are solved in its implementation widen the gap between the original system and its implementation. Here we try to solve these problems inside the proof-theoretic system by avoiding global rules.

In this chapter we will present a system for first-order predicate logic in the calculus of structures which is local except for the treatment of variables. The propositional fragment is a fully local system.

3.1 Atomic Forms

The **atomic form** of an inference rule is the rule that occurs if we restrict the active formulas to atoms. For example, consider the w_R rule of GS1:

$$\mathrm{w}_R \frac{\vdash \Phi}{\vdash \Phi, A}$$

Its atomic form would be:

$$\mathrm{w}_R \frac{\vdash \Phi}{\vdash \Phi, P}$$

In the general form of the rule A denotes an arbitrary formula. In the atomic form A is replaced by P, which usually denotes an atomic formula.

Locality is mostly achieved by reducing the problematic rules to their atomic forms. We say that a rule can be **reduced to atomic** if it can be replaced by its atomic form without losing anything in provability. For example, the Axiom rule of GS1 can be reduced to atomic form:

$$\mathrm{Ax} \frac{}{\vdash A, \bar{A}} \quad \text{is equivalent to} \quad \mathrm{Ax} \frac{}{\vdash P, \bar{P}}$$

where P is an atomic formula. This is proved by showing that any general instance of Ax can be derived for the atomic version of Ax. The atomic forms are of course local rules since they only need to duplicate, erase or compare atoms.

In the following we will show that identity, cut and weakening in SKSgq are equivalent to their atomic forms. Contraction requires a special treatment which will be presented in the next section.

Identity and Cut

The identity rule $\mathrm{i}{\downarrow} \dfrac{S\{t\}}{S[R, \bar{R}]}$ can be reduced to its atomic form, the **atomic identity** rule $\mathrm{ai}{\downarrow} \dfrac{S\{t\}}{S[\alpha, \bar{\alpha}]}$. We will prove this by induction on the complexity of the active formulas:

- The base case is when R is an atom or a unit. In the first case the instance of the general rule is also an instance of its atomic form. For the second case, assume $R = t$. Then $\bar{R} = f$ and the given identity instance is an instance of equivalence:

$$= \frac{t}{[t, f]}$$

- Now assume $R = [P, Q]$ and the induction hypothesis holds for P, Q. This means that instances of identity that introduce P or Q can be derived for atomic identity. Then we have:

$$\mathrm{i}{\downarrow}\frac{S\{t\}}{S[P, Q, (\bar{P}, \bar{Q})]} \quad \rightsquigarrow \quad \mathrm{i}{\downarrow}\frac{=\dfrac{\mathrm{i}{\downarrow}\dfrac{S\{t\}}{S[Q, \bar{Q}]}}{S([Q, \bar{Q}], t)}}{\mathrm{s}\dfrac{\mathrm{s}\dfrac{S([Q, \bar{Q}], [P, \bar{P}])}{S[Q, ([P, \bar{P}], \bar{Q})]}}{S[P, Q, (\bar{P}, \bar{Q})]}}$$

  If R is a conjunction we use the same derivation. Just assume that $R = (\bar{P}, \bar{Q})$.

- If $R = \exists x T$ and the induction hypothesis holds for T then:

$$\mathrm{i}{\downarrow}\frac{S\{t\}}{S[\exists x T, \forall x \bar{T}]} \quad \rightsquigarrow \quad \mathrm{i}{\downarrow}\frac{=\dfrac{S\{t\}}{S\{\forall x t\}}}{\mathrm{u}{\downarrow}\dfrac{S\{\forall x [T, \bar{T}]\}}{S[\exists x T, \forall x \bar{T}]}}$$

  The same derivation works in the case that $R = \forall x \bar{T}$.

As we already mentioned, the same is true for the Axiom rule of the sequent calculus. What is new here is that the cut rule $\mathrm{i}{\uparrow}\dfrac{S(R, \bar{R})}{S\{f\}}$ can also be reduced to the **atomic cut** rule $\mathrm{ai}{\uparrow}\dfrac{S(\alpha, \bar{\alpha})}{S\{f\}}$ because of the duality to i↓. We prove it inductively considering the dual derivations:

- If R is an atom then the instance of the general rule is also an instance of its atomic form. If R is a unit, let's say $R = t$, then $\bar{R} = f$ and the given cut instance is an instance of equivalence:

$$= \frac{[t, f]}{f}$$

- If $R = [P, Q]$ and the induction hypothesis holds for P, Q then:

$$\mathrm{i}\uparrow \frac{S([\bar{P}, \bar{Q}], [P, Q])}{S\{f\}} \quad \rightsquigarrow \quad \mathrm{s}\,\frac{S([\bar{P}, \bar{Q}], [P, Q])}{\mathrm{s}\,\frac{S([\bar{Q}, [(\bar{P}, P), Q])}{\mathrm{i}\uparrow \frac{S[(Q, \bar{Q}), (P, \bar{P})]}{= \frac{S[(Q, \bar{Q}), f]}{\mathrm{i}\uparrow \frac{S(Q, \bar{Q})}{S\{f\}}}}}}$$

If R is a conjunction we use the same derivation.

- If $\exists x T$ then:

$$\mathrm{i}\uparrow \frac{S(\exists x T, \forall x \bar{T})}{S\{f\}} \quad \rightsquigarrow \quad \mathrm{u}\uparrow \frac{S(\exists x T, \forall x \bar{T})}{\mathrm{i}\uparrow \frac{S\{\exists x (T, \bar{T})\}}{= \frac{S\{\exists x f\}}{S\{f\}}}}$$

The same derivation works in the case that R is of the shape $\forall x Q$. Just consider $Q = \bar{T}$.

The Cut rule of GS1 can not be replaced by its atomic form. Of course Cut is admissible for the sequent calculus. This means that cuts in a proof can trivially be reduced to atomic by eliminating them. However, this works only for proofs, i.e. derivations with axioms as premisses. Cut elimination does not work for arbitrary deductions. In the calculus of structures, the reduction works for any derivation. This is due to the perfect duality between identity and cut, which allows us to dualise the reduction of i↓ and get a reduction of i↑. The fact that we can reduce cuts to atomic is a very important feature of CoS. It allows for a much simpler analysis during cut elimination.

Weakening

The weakening $\mathrm{w}\!\downarrow \dfrac{S\{f\}}{S\{R\}}$ can also be reduced to **atomic weakening** $\mathrm{aw}\!\downarrow \dfrac{S\{f\}}{S\{\alpha\}}$. This is again done by inductively replacing a general instance of weakening by instances on smaller formulas:

- The base cases are when R is an atom or a unit. In the first case we have an atomic instance of w↓, thus an instance of aw↓. In the second case we have two subcases. If $R = f$ then the premise and the conclusion of w↓ are the same so we can unify them. If $R = t$ then we replace the instance with the following derivation:

$$\mathrm{w}\!\downarrow \frac{S\{f\}}{S\{t\}} \qquad \rightsquigarrow \qquad \begin{array}{r} = \dfrac{S\{f\}}{S(t,f)} \\ = \dfrac{S([t,t],f)}{} \\ \mathrm{s}\ \dfrac{S[t,(t,f)]}{} \\ = \dfrac{}{S\{t\}} \end{array}$$

- Assume that $R = [P,Q]$ and the induction hypothesis holds for P, Q. So we can apply w↓ for them taking for granted that this general instance can be derived for the atomic:

$$\mathrm{w}\!\downarrow \frac{S\{f\}}{S[P,Q]} \qquad \rightsquigarrow \qquad \begin{array}{r} = \dfrac{S\{f\}}{S[f,f]} \\ \mathrm{w}\!\downarrow \dfrac{S[f,Q]}{} \\ \mathrm{w}\!\downarrow \dfrac{}{S[P,Q]} \end{array}$$

- If $R = (P,Q)$ then:

$$\mathrm{w}\!\downarrow \frac{S\{f\}}{S(P,Q)} \qquad \rightsquigarrow \qquad \begin{array}{r} = \dfrac{S\{f\}}{S(f,f)} \\ \mathrm{w}\!\downarrow \dfrac{S(f,Q)}{} \\ \mathrm{w}\!\downarrow \dfrac{}{S(P,Q)} \end{array}$$

- If $R = \exists x P$ then:

$$\mathrm{w}\!\downarrow \frac{S\{f\}}{S\{\exists x P\}} \qquad \rightsquigarrow \qquad \begin{array}{r} = \dfrac{S\{f\}}{S\{\exists x f\}} \\ \mathrm{w}\!\downarrow \dfrac{}{S\{\exists x P\}} \end{array}$$

- If $R = \forall x P$ then:

$$\mathrm{w}\!\downarrow \frac{S\{f\}}{S\{\forall x P\}} \qquad \rightsquigarrow \qquad \begin{array}{c} = \dfrac{S\{f\}}{S\{\forall x f\}} \\ \mathrm{w}\!\downarrow \dfrac{}{S\{\forall x P\}} \end{array}$$

The reduction of $\mathrm{w}\!\uparrow \dfrac{S\{R\}}{S\{t\}}$ to the atomic form $\mathrm{aw}\!\uparrow \dfrac{S\{\alpha\}}{S\{t\}}$ can be easily taken from the dual derivations.

3.2 Atomic Contraction

Unfortunately, we can not do the same thing with contraction. It can not be reduced to atomic form in SKSgq. By this we mean that if we just replace $\mathrm{c}{\downarrow}\dfrac{S[R,R]}{S\{R\}}$ by its atomic version $\mathrm{ac}{\downarrow}\dfrac{S[\alpha,\alpha]}{S\{\alpha\}}$, called **atomic contraction**, we will not be able to derive any general instance of contraction.

$$\mathrm{I}_1{\downarrow}\frac{S[\exists xR,\exists xT]}{S\{\exists x[R,T]\}} \qquad \mathrm{I}_1{\uparrow}\frac{S\{\forall x(R,T)\}}{S(\forall xR,\forall xT)}$$

$$\mathrm{m}\frac{S[(R,U),(T,V)]}{S([R,T],[U,V])}$$

$$\mathrm{I}_2{\downarrow}\frac{S[\forall xR,\forall xT]}{S\{\forall x[R,T]\}} \qquad \mathrm{I}_2{\uparrow}\frac{S\{\exists x(R,T)\}}{S(\exists xR,\exists xT)}$$

Figure 3.1: The Medials

For this we will need the **medials** (Figure 3.1). These rules are sound. We can easily derive them for c↓ and w↓:

$$\mathrm{m}\frac{S[(R,U),(T,V)]}{S([R,T],[U,V])} \quad \rightsquigarrow \quad \cfrac{\cfrac{\cfrac{\cfrac{\mathrm{w}{\downarrow}\dfrac{S[(R,U),(T,V)]}{S[(R,U),(T,[U,V])]}}{\mathrm{w}{\downarrow}\dfrac{S[(R,U),([R,T],[U,V])]}{S[(R,[U,V]),([R,T],[U,V])]}}}{\mathrm{w}{\downarrow}\dfrac{S[([R,T],[U,V]),([R,T],[U,V])]}{S([R,T],[U,V])}}}{\mathrm{c}{\downarrow}\dfrac{S[([R,T],[U,V]),([R,T],[U,V])]}{S([R,T],[U,V])}}}$$

$$\mathrm{I}_1{\downarrow}\frac{S[\exists xR,\exists xT]}{S\{\exists x[R,T]\}} \quad \rightsquigarrow \quad \cfrac{\cfrac{\mathrm{w}{\downarrow}\dfrac{S[\exists xR,\exists xT]}{S[\exists xR,\exists x[R,T]]}}{\mathrm{w}{\downarrow}\dfrac{S[\exists x[R,T],\exists x[R,T]]}{S\{\exists x[R,T]\}}}}{\mathrm{c}{\downarrow}\dfrac{S[\exists x[R,T],\exists x[R,T]]}{S\{\exists x[R,T]\}}}$$

$$\mathrm{I}_2\downarrow\frac{S[\forall xR,\forall xT]}{S\{\forall x[R,T]\}}\qquad\rightsquigarrow\qquad\begin{array}{c}\mathrm{w}\downarrow\dfrac{S[\forall xR,\forall xT]}{S[\forall xR,\forall x[R,T]]}\\ \mathrm{w}\downarrow\dfrac{S[\forall xR,\forall x[R,T]]}{S[\forall x[R,T],\forall x[R,T]]}\\ \mathrm{c}\downarrow\dfrac{S[\forall x[R,T],\forall x[R,T]]}{S\{\forall x[R,T]\}}\end{array}$$

Their duals $\mathrm{I}_1\uparrow$ and $\mathrm{I}_2\uparrow$ are of course given by the dual derivations. The rule m is a self-dual rule. In the next section we will show that these rules are local. This will be achieved through a graph-theoretic approach.

Now, if we admit the medials, we are able to reduce contraction $\mathrm{c}\downarrow\dfrac{S[R,R]}{S\{R\}}$ to atomic form:

- The base cases first. If R is an atom then the instance of the general rule is also an instance of its atomic form. If R is a unit, let's say $R=f$, then the given instance is an instance of equivalence:

$$=\frac{S[f,f]}{S\{f\}}$$

  The same holds if $R=t$.

- If $R=[P,Q]$ and the induction hypothesis holds for P,Q then:

$$\mathrm{c}\downarrow\frac{S[[P,Q],[P,Q]]}{S[P,Q]}\qquad\rightsquigarrow\qquad\begin{array}{c}=\dfrac{S[[P,Q],[P,Q]]}{S[P,P,Q,Q]}\\ \mathrm{c}\downarrow\dfrac{S[P,P,Q,Q]}{S[P,P,Q]}\\ \mathrm{c}\downarrow\dfrac{S[P,P,Q]}{S[P,Q]}\end{array}$$

- If $R=(P,Q)$ then:

$$\mathrm{c}\downarrow\frac{S[(P,Q),(P,Q)]}{S(P,Q)}\qquad\rightsquigarrow\qquad\begin{array}{c}\mathrm{m}\dfrac{S[(P,Q),(P,Q)]}{S([P,P],[Q,Q])}\\ \mathrm{c}\downarrow\dfrac{S([P,P],[Q,Q])}{S([P,P],Q)}\\ \mathrm{c}\downarrow\dfrac{S([P,P],Q)}{S(P,Q)}\end{array}$$

- If $R=\forall xP$ then:

$$\mathrm{c}\downarrow\frac{S[\forall xP,\forall xP]}{S\{\forall xP\}}\qquad\rightsquigarrow\qquad\begin{array}{c}\mathrm{I}_2\downarrow\dfrac{S[\forall xP,\forall xP]}{S\{\forall x[P,P]\}}\\ \mathrm{c}\downarrow\dfrac{S\{\forall x[P,P]\}}{S\{\forall xP\}}\end{array}$$

- If $R = \exists x P$ then:

$$\mathrm{c}{\downarrow}\,\frac{S[\exists x P, \exists x P]}{S\{\exists x P\}} \qquad \rightsquigarrow \qquad \begin{array}{l}\mathrm{I}_1\,{\downarrow}\,\dfrac{S[\exists x P, \exists x P]}{S\{\exists x[P, P]\}}\\ \mathrm{c}{\downarrow}\,\dfrac{S\{\exists x[P, P]\}}{S\{\exists x P\}}\end{array}$$

The reduction of co-contraction $\mathrm{c}{\uparrow}\,\dfrac{S\{R\}}{S(R, R)}$ to the atomic form $\mathrm{ac}{\uparrow}\,\dfrac{S\{\alpha\}}{S(\alpha, \alpha)}$ is taken by dualising the reduction above.

Contraction in the Sequent Calculus

In the sequent calculus we can reduce the axioms to atomic and eliminate all instances of cut. However, contraction can not be treated locally. In Gentzen's sequent system G3c contraction is admissible, in the sense that it not explicit but comes as a property of the system. Unfortunately, G3c has a context-sharing $\wedge_R$ rule, which is non-local.

It mains to see if we can reduce contraction in GS1 to atomic form without losing completeness. The answer is negative, as Brünnler showed with his counter-example in [2]. It works for various sequent systems, as long as they have a multiplicative $\wedge_R$ rule:

Theorem 3.2.1. *The following sequent:*

$$\vdash \alpha \wedge b, (\bar{\alpha} \vee \bar{b}) \wedge (\bar{\alpha} \vee \bar{b})$$

is valid but it has no proof in multiplicative GS1 in which all contractions are atomic.

Proof. The sequent is obviously valid. This can be easily shown with a truth table. It contains no atoms, so atomic contraction cannot be applied. Each applicable rule leads to a premise that is not valid. So there is no proof of it. □

Thus, deep inference is necessary for reducing contraction to atomic and obtaining locality. We can not construct a local system in the sequent calculus, not even for the propositional logic.

3.3 System SKSq

We now obtain system SKSq, shown in Figure 3.2. It occurs from SKSgq by restricting identity, cut, weakening and contraction to atomic form and adding the medials. The up-fragment is admissible, as in SKSgq. The asymmetric system KSq consists of the down-fragment. The propositional fragment SKS does not contain u↓,n↓,I_1 ↓,I_2 ↓ and their duals. The De Morgan laws now become useless. Observe that the rules in SKSq introduce negation only on atoms. Thus, it is possible to restrict negation to atoms from the beginning, considering formulas in negation normal form, and drop the equations for negation entirely. This is customary in the one-sided sequent calculus.

From the equivalences between general and atomic inference rules shown in Sections 3.1 and 3.2 we obtain the following theorem:

Theorem 3.3.1. *System SKSq and system SKSgq are strongly equivalent.*

Thus, the correspondence with the sequent calculus holds for SKSq. It is a sound and complete deductive system for classical logic. The same of course holds for the asymmetric systems:

Theorem 3.3.2. *System KS and system KSg are strongly equivalent.*

We often use the general versions of the rules, instead of the atomic, inside a derivation in SKS. For example, we write:

$$\mathrm{c}{\downarrow}\,\frac{S[(\alpha,\beta),(\alpha,\beta)]}{S(\alpha,\beta)}$$

and we mean:

$$\begin{array}{r} \mathrm{m}\,\dfrac{S[(\alpha,\beta),(\alpha,\beta)]}{S([\alpha,\alpha],[\beta,\beta])} \\ \mathrm{ac}{\downarrow}\,\dfrac{S([\alpha,\alpha],[\beta,\beta])}{S([\alpha,\alpha],\beta)} \\ \mathrm{ac}{\downarrow}\,\dfrac{S([\alpha,\alpha],\beta)}{S(\alpha,\beta)} \end{array}$$

We do that for simplicity and shorter proofs.

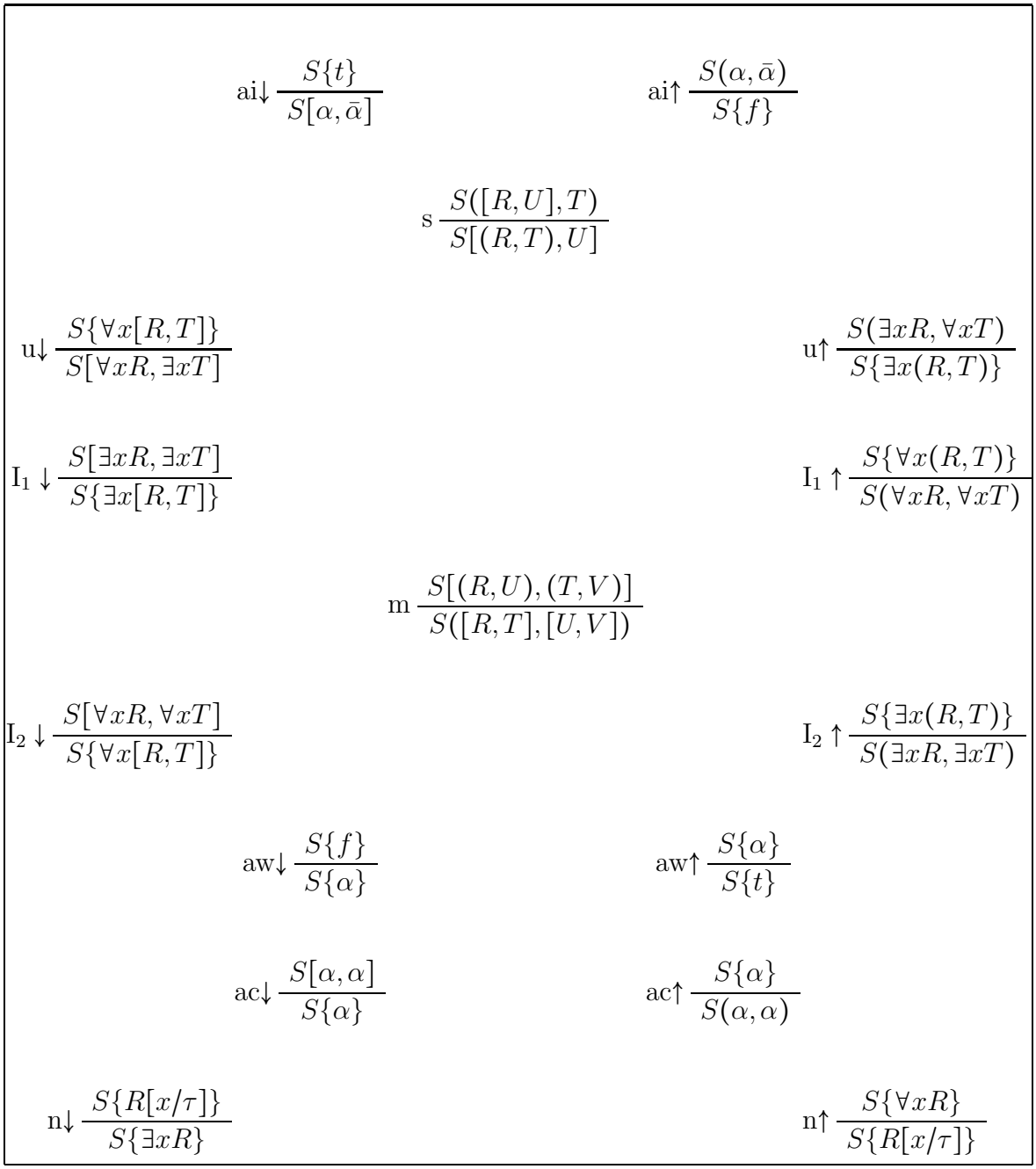
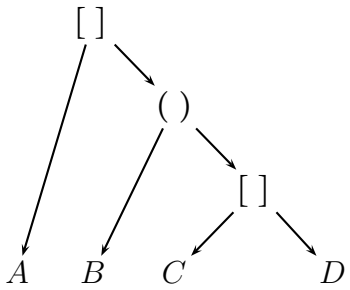


Figure 3.2: System SKSq

We have not said anything yet about the locality of the switch, the medials, the equivalence rules and the rules for the quantifiers. The switch rule involves formulas of unbounded size, but it does not require inspecting them. To see this, consider formulas represented as trees. For example, the formula $[A, (B, [C, D])]$ is represented as follows:



The switch rule:

$$\mathrm{s}\,\frac{S([R,U],T)}{S[(R,T),U]}$$

can be implemented by changing the marking of two nodes and exchanging two pointers (Figure 3.3). The same technique works for the medials (Figure 3.4). We see that the concept of locality depends on the representation of formulas. Some rules are local when formulas are represented as trees, but they are not when formulas are represented as strings.

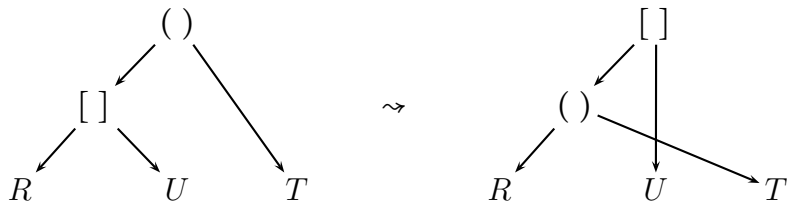


Figure 3.3: Locality of the Switch

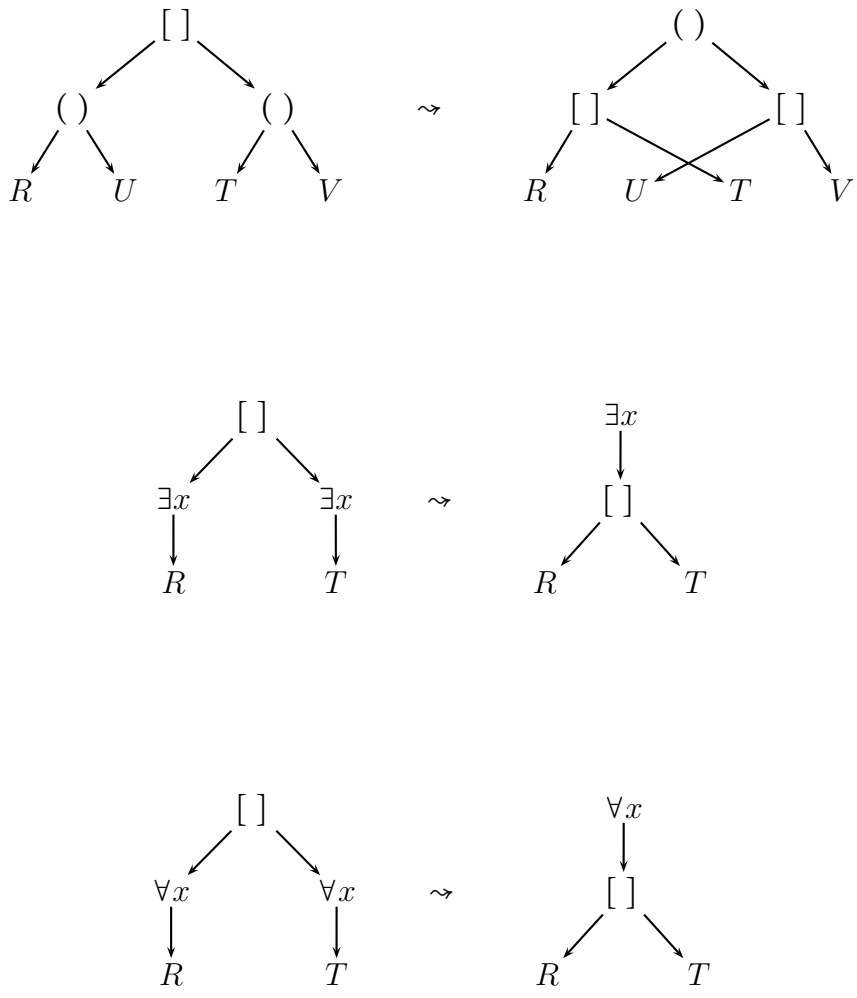


Figure 3.4: Locality of the Medials

The equivalence relations that have to do with the propositional part are local as well. However, the proviso in the following equivalence rule makes it non-local:

$$\forall y R = \exists y R = R, \text{ where } y \text{ is not free in } R.$$

To add or remove a quantifier, a formula of unbounded size has to be checked for occurrences of the variable y. The same holds for the variable renaming equation.

The second source of non-locality hides in the predicate rules. While u↓ is local (Figure 3.5), the n↓ rule demands unbounded inspection of the formula R. A term τ of unbounded size has to be copied into an unbounded number of occurrences of x in R. The unbounded size of τ can be dealt with by introducing only one function symbol at every instance. However, the unbounded number of the occurences of the variable makes SKSq non-local.

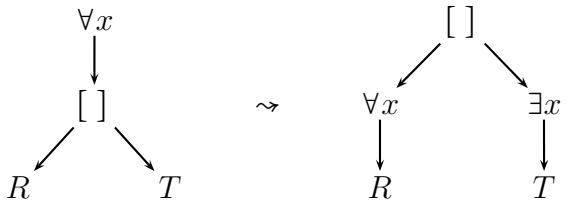


Figure 3.5: Locality of the Universal

Therefore, the propositional fragment SKS, with the corresponding equivalence rules, is a local system. No rule needs to inspect formulas of unbounded size. System SKSq is not fully local because of the treatment of variables and the n↓ rule.

Chapter 4

Normalisation

Formalisms entail some notion of **normalisation** inside their proof systems. By this we mean a transformation of deductions into ones with desirable properties. Usually, a normal proof is considered as a representative of a class of equivalent proofs. Natural deduction systems possess a notion of normalisation that removes all redundancies from a proof. This procedure is stongly connected with computation. In the sequent calculus, normalisation means cut elimination. The existence of cut-free proofs makes the proof search finite. In this chapter we will present several notions of normalisation for the calculus of structures.

4.1 Normalisation in the Traditional Formalisms

Before we see normalisation techniques for CoS it would be useful to give a brief description of normalisation in natural deduction and sequent calculus, two widely used shallow formalisms.

Normalisation in the Natural Deduction

The Hilbert-style axiomatization of deductive reasoning caused great dissatisfaction to many logicians, who would prefer a more natural treatment of logic. The result of their attempts was **natural deduction**. It is a proof calculus in which logical reasoning is expressed by inference rules that come as close as possible to actual reasoning. In Figure 4.1 we see a system for classical first-order logic. For every connective and quantifier there is an

introduction and an **elimination** rule. The rule $\perp_c$ is the **classical absurdity** rule. It is an explicit form of the scheme of proof by contradiction. The $\rightarrow_E$ rule is same with the modus ponens of Hc. Derivations have the form of deduction trees, as in sequent calculus. Formulas appearing at the top nodes, the leaves, are the **premisses**. They can be either **closed** (in [ ]) or **open**. When you start building a derivation your assumptions are open. But after the application of some certain rules, a set of premisses may become closed. This happens in rules $\vee_E$,$\rightarrow_I$, $\exists_E$ and $\perp_c$. For example, in the $\rightarrow_I$ case, we have a derivation:

$$\begin{array}{c} A \\ \vdots \\ B \end{array}$$

It corresponds to the implication $A \rightarrow B$. If we apply $\rightarrow_I$ we will have as conclusion $A \rightarrow B$. So it is quite natural to close the open assumption A since it is going to appear as a premise in the conclusion:

$$\rightarrow_I \frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \rightarrow B}$$

By the way, this inference rule is the analogue of the deduction theorem that we saw for other formalisms. The other cases of closed premisses happen for similar reasons. A formula A is valid if there is a **proof** of A, a deduction of A with all premisses closed. For example, here is a proof of the classically valid formula $\neg\forall x\neg A(x) \rightarrow \exists x A(x)$:

$$\rightarrow_I, w \frac{\perp_c, u \dfrac{\rightarrow_E \dfrac{[\neg\forall x\neg A(x)]^w \qquad \forall_I \dfrac{\rightarrow_I, v \dfrac{\rightarrow_E \dfrac{[\neg\exists x A(x)]^u \qquad \exists_I \dfrac{[A(x)]^v}{\exists x A(x)}}{\perp}}{\neg A(x)}}{\forall x\neg A(x)}}{\perp}}{\exists x A(x)}}{\neg\forall x\neg A(x) \rightarrow \exists x A(x)}$$

The variable y appearing in $\forall_I$ and $\exists_E$ is called a **proper** variable. It has not to be free in A, unless $y \equiv x$, not to be free in the open assumptions, except of course $A[x/y]$, and not to be free in C (for $\exists_E$).

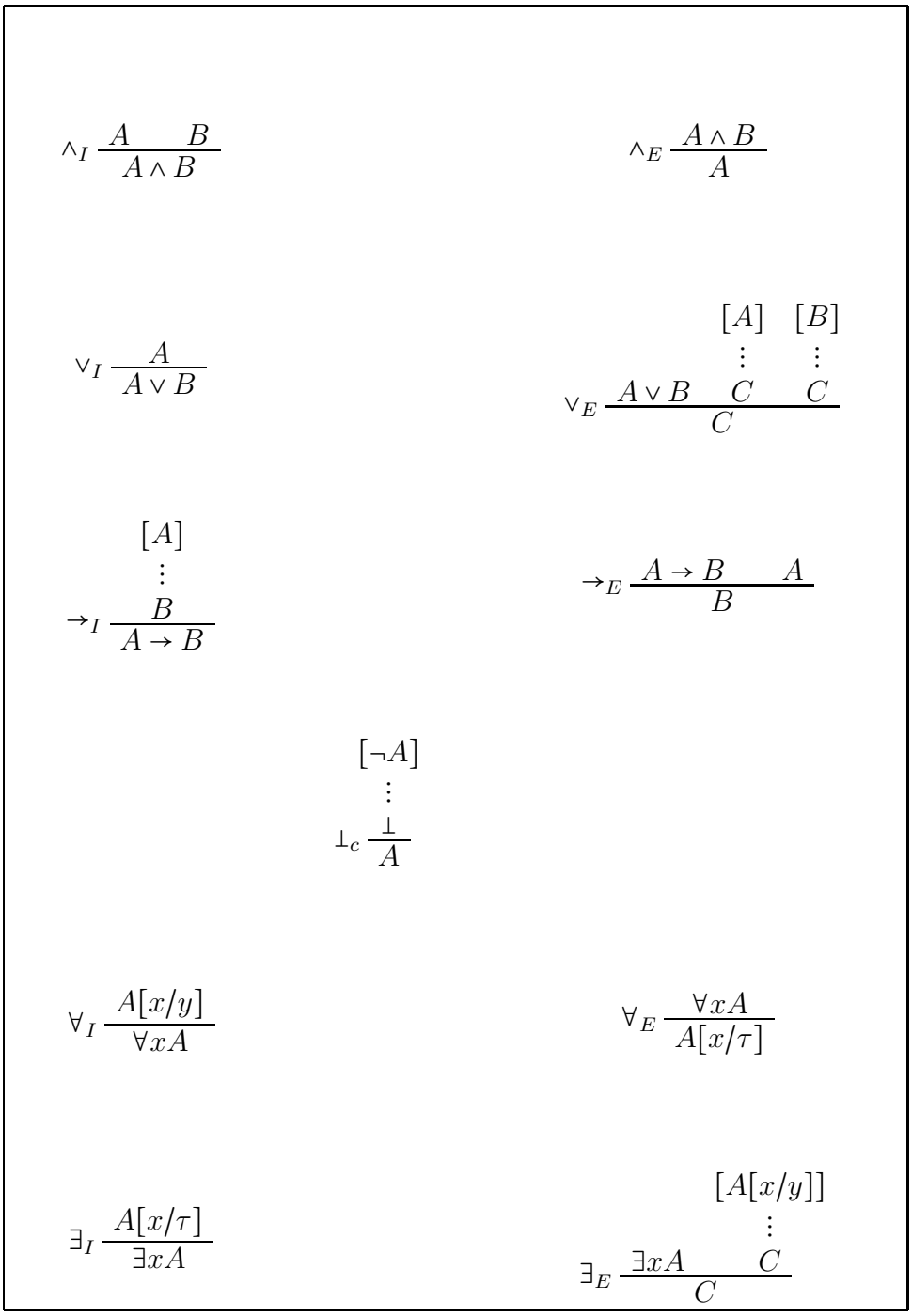


Figure 4.1: System Nc

A succession of the introduction and the elimination rule of a certain connective in a derivation is called a **redex**. Normalisation aims at removing those detours. Let us see the possible cases and their **contractums**, i.e. their normalisations:

$$\dfrac{\dfrac{\begin{matrix}D_1\\A_1\end{matrix}\quad\begin{matrix}D_2\\A_2\end{matrix}}{A_1 \wedge A_2}\wedge_I}{A_i}\wedge_E \quad \rightsquigarrow \quad \begin{matrix}D_i\\A_i\end{matrix}$$

$$\dfrac{\begin{matrix}D\\A_i\end{matrix}}{A_1 \vee A_2}\vee_I \quad \dfrac{\begin{matrix}[A_1]^u\\D_1\\C\end{matrix}\quad\begin{matrix}[A_2]^v\\D_2\\C\end{matrix}}{C}\vee_{E,u,v} \quad \rightsquigarrow \quad \begin{matrix}D\\A_i\\D_i\\C\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}[A]^u\\D_1\\B\end{matrix}}{A \to B}\to_{I,u} \quad \begin{matrix}D_2\\A\end{matrix}}{B}\to_E \quad \rightsquigarrow \quad \begin{matrix}D_2\\A\\D_1\\B\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}D\\A\end{matrix}}{\forall y A[x/y]}\forall_I}{A[x/t]}\forall_E \quad \rightsquigarrow \quad \begin{matrix}D[x/t]\\A[x/t]\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}D\\A[y/t]\end{matrix}}{\exists x A[y/x]}\exists_I \quad \begin{matrix}[A]^u\\D'\\C\end{matrix}}{C}\exists_{E,u} \quad \rightsquigarrow \quad \begin{matrix}D\\A[y/t]\\D'[y/t]\\C\end{matrix}$$

Each redex consists of the introduction and elimination of a logical symbol. These are short redexes and easily eliminated, as we can see above. However, an introduced formula may be used as a minor premise of an application of $\vee_E$ or $\exists_E$, then stay the same throughout a sequence of applications of these rules, being eliminated at the end. This is also a redex and we have to use permutations of rules in order to reduce it to the simple cases. For a

detailed analysis look at [1]. For every derivation there is a derivation with the same premisses and conclusion and without redexes, called a **normal form**. An interesting point is that in a normal derivation all eliminations happen above introductions. The rate of growth of a derivation under normalisation is hyperexponential. This means that, if a derivation consists of n rule instances, its normal form could have length equal to $c \cdot e^n$.

The most interesting aspect of natural deduction is its close correspondence to the simply typed **$\boldsymbol{\lambda}$-calculus**. This is a model of computation introduced by Alonzo Church in the 1930s. The syntax of **$\boldsymbol{\lambda}$-terms** is the following:

- A variable x is a λ-term.
- If t is a λ-term, and x is a variable, then $\lambda x.t$ is a λ-term, called a **$\boldsymbol{\lambda}$-abstraction**. The λ is said to bind x in t.
- If t and s are λ-terms, then $(t)s$ is a λ-term, called the **application** of t on s.

The **free variables** of a term are those variables not bound by a lambda abstraction. We also use metavariables for terms, like t, u, v. Intuitively, a λ-term corresponds to a function or a program. A λ-abstraction $\lambda x.t$ represents a function that takes a single input and an application $(t)s$ represents the application of function t to some input s. For example, $\lambda x.x$ represents the identity function and $\lambda x.y$ represents the constant function that always returns y, no matter the input.

So far we have defined the untyped λ-calculus. Now we can define the **types**. We begin by fixing a set of primitive types B. These are called **atomic** types. We can imagine them as ordinary data types, such as $\mathbb{NAT}$ or $\mathbb{BOOL}$. The syntax of types is:

- An atomic type is a type.
- If τ and σ are types then $\tau \to \sigma$ is a type.

Types are assigned to λ-terms by induction on the compexity of their construction:

- A variable x^A is a λ-term of type A.
- If t is a λ-term of type B, under the hypothesis that the free variable x^A is of type A, then $\lambda x^A.t$ is a term of type $A \to B$ and the condition for x^A vanishes. This can be written as an inference rule:

$$\to_{I,\,u} \frac{\begin{array}{c}[x^A : A]^u \\ \vdots \\ t : B\end{array}}{\lambda x^A.t : A \to B}$$

- If t and s are λ-terms of types $A \to B$ and A respectively, then $(t)s$ is a lambda term of type B. This can be written:

$$\to_E \frac{t : A \to B \qquad s : A}{(t)s : B}$$

Thus, a typed λ-term can be represented by a deduction having as conclusion the term with its type. Closed hypotheses correspond to the arguments of the program and open hypotheses just denote free variables inside it.

Logic	Programming
Formulas	Types
Proofs	Programs
Normalisation	Computation

Figure 4.2: Curry-Howard Correspondence

If we consider types as formulas and ignore the terms, the typing rules $\to_I$ and $\to_E$ introduced above are identical to the rules for $\to$ of natural deduction. We can observe a syntactic analogy between derivations and terms (programs) of the typed lambda calculus, since they are both built from the same inference rules. A typed term can be seen as the proof of its type (seen as a formula). A proof is a program and the formula it proves is the return type of the program. Thus, the given computation model is equivalent to $\to$Nm, the fragment of Nc containing only $\to_I$ and $\to_E$. We can define more type constructors, besides $\to$, that will correspond to the connectives $\vee$ and $\wedge$.

Furthermore, it can be shown that the normalisation of proofs in →Nm corresponds to computing in the simply typed λ-calculus. Computation is expressed by **β-reduction**:

$$(\lambda x.t)s \quad \longrightarrow \quad t[x/s]$$

An expression of the form $(\lambda x.t)s$ is called a **redex** and $t[x/s]$ is its **contractum**. Beta reduction says the obvious: if a function t with argument x is applied to s then every occurence of x in t will be replaced by s. Check that this rewriting rule is exactly the analogue of:

$$\dfrac{\dfrac{\begin{array}{c}[A]^u \\ D_1 \\ B\end{array}}{A \to B}\,{\scriptstyle \to_I,\,u} \qquad \begin{array}{c} D_2 \\ A \end{array}}{B}\,{\scriptstyle \to_E} \qquad \rightsquigarrow \qquad \begin{array}{c} D_2 \\ A \\ D_1 \\ B \end{array}$$

In other words, the correspondence described above respects the redex - contractum relation. It is interesting that **plugging** a proof on the premisses of a derivation corresponds to substitution, which can express computation. The close connection between deductive systems and computational machines is known as the **Curry-Howard correspondence** (Figure 4.2). Designers of functional programming languages prefer natural deduction because of that close correspondence with term calculi.

Cut Elimination in the Sequent Calculus

We are going to prove that the Cut rule is admissible. This means that if there is a proof of $\Gamma \vdash \Delta$ in GS1 + Cut then there is a proof of it in GS1. A **normal** proof in the sequent calculus is a proof without cuts. Let us recall the rule:

$$\text{Cut}\,\dfrac{\vdash \Phi, A \qquad \vdash \Psi, \bar{A}}{\vdash \Phi, \Psi}$$

So we have to show that if there is a cut-free proof of $\vdash \Phi, A$ and a cut-free proof of $\vdash \Psi, \bar{A}$ then there is a cut-free proof of $\vdash \Phi, \Psi$ (closure under Cut).

First we need to give the following definitions:

Definition 4.1.1. The **level** of a cut is the sum of the depths of the deductions of its premisses.

The level is a measure of how deep a cut is inside a proof.

Definition 4.1.2. The **rank** of a cut is defined as $|A| + 1$, where $|A|$ is the depth of the tree representation of the formula.
The **cutrank** of a deduction is the maximum of the ranks of the cut formulas occuring in it.

The rank of a cut is a measure of the complexity of the cut formula A.

We will give a brief description of Gentzen's algorithm for eliminating cuts from a given proof. The method proceeds by a main induction on the cutrank with a subinduction on the level of the cut. In other words, the order in which cuts are removed is not random. Each time we choose the topmost cut among all cuts with rank equal to the rank of the whole deduction. Then the cut is pushed upwards to the top of the proof using permutations of rules. When it meets the axioms it gets eliminated. Then we choose the next cut etc.

Let's give a more detailed description. In the conclusion of each rule, the formula not in the context is called the **principal** formula. In addition, we will consider axioms of the form:

$$\text{Ax}\ \frac{}{\vdash \Gamma, P, \bar{P}}$$

where P is an atomic formula. This is an equivalent approach, since the Ax rule can be reduced to atomic and putting a context Γ in it just makes the weakening rule implicit (admissible). This happens in systems G2c and G3c (see [1]). Consider a topmost maximal-rank cut inside a proof:

$$\text{Cut}\ \frac{\begin{array}{cc}\Delta_1 & \Delta_2 \\ \vdash \Phi, A & \vdash \Psi, \bar{A}\end{array}}{\vdash \Phi, \Psi}$$

where Δ_1, Δ_2 are cut-free proofs. We have to examine the following cases:

1. One of the premisses is an axiom instance. We have to examine several subcases. Let's see the main two:

   - The premise on the left is an application of Ax and the cutformula is not principal:

$$\text{Cut}\ \frac{\begin{array}{cc}\text{Ax}\ \dfrac{}{\vdash \Gamma, P, \bar{P}, A} & D_1 \\ & \vdash \Gamma', \bar{A}\end{array}}{\vdash \Gamma, \Gamma', P, \bar{P}}$$

In this case the conclusion is an axiom, so we can take it as the cut-free proof.

- The premise on the left is an application of Ax and the principal formula is also a cutformula. In that case we could just apply all the weakenings needed to D_1:

$$\dfrac{\dfrac{}{\vdash \Gamma, P, \bar{P}}\,\text{Ax} \qquad \begin{matrix} D_1 \\ \vdash \Gamma', \bar{P} \end{matrix}}{\vdash \Gamma, \Gamma', \bar{P}}\,\text{Cut}$$

$$\Downarrow$$

$$\dfrac{\begin{matrix} D_1 \\ \vdash \Gamma', \bar{P} \end{matrix}}{\vdash \Gamma, \Gamma', \bar{P}}\,\text{w}_R$$

When the cut meets the axiom they both vanish. This phenomenon is often referred to as **interaction**. It exhibits the symmetry between Ax and Cut.

2. The premisses are not axioms but in at least one of them the cut formula is not principal. In that case we can permute the cut upwards, passing it over the rules that do not act on the cut formula.

3. The cut formula is principal on both sides. In this case, on one branch a logical rule applies to the main connective of the cut formula and on the other branch the corresponding rule applies to the dual connective of the dual cut formula. If the cut formula is a conjunction (or a disjunction) we make the following transformation:

$$\dfrac{\dfrac{\begin{matrix} D_1 \\ \vdash \Gamma, A \end{matrix} \qquad \begin{matrix} D_2 \\ \vdash \Gamma', B \end{matrix}}{\vdash \Gamma, \Gamma', A \wedge B}\,\wedge_R \qquad \dfrac{\begin{matrix} D_3 \\ \vdash \Delta, \bar{A}, \bar{B} \end{matrix}}{\vdash \Delta, \bar{A} \vee \bar{B}}\,\vee_R}{\vdash \Gamma, \Gamma', \Delta}\,\text{Cut}$$

$$\Downarrow$$

$$\dfrac{\begin{matrix} D_1 \\ \vdash \Gamma, A \end{matrix} \qquad \dfrac{\begin{matrix} D_2 \\ \vdash \Gamma', B \end{matrix} \qquad \begin{matrix} D_3 \\ \vdash \Delta, \bar{A}, \bar{B} \end{matrix}}{\vdash \Gamma', \Delta, \bar{A}}\,\text{Cut}}{\vdash \Gamma, \Gamma', \Delta}\,\text{Cut}$$

We see that we traded the cut for two cuts of lower rank and lower level. About the same happens in the case where the cut formula is existentially (or universally) quantified:

$$\dfrac{\forall_R \dfrac{\begin{matrix}D_1\\ \vdash \Gamma, A[x/y]\end{matrix}}{\vdash \Gamma, \forall x A} \qquad \exists_R \dfrac{\begin{matrix}D_2\\ \vdash \Gamma', \overline{A[x/t]}\end{matrix}}{\vdash \Gamma', \exists x \bar{A}}}{\vdash \Gamma, \Gamma'}\,\text{Cut}$$

$$\Downarrow$$

$$\dfrac{\begin{matrix}D_1[y/t]\\ \vdash \Gamma, A[x/t]\end{matrix} \qquad \begin{matrix}D_2\\ \vdash \Gamma', \overline{A[x/t]}\end{matrix}}{\vdash \Gamma, \Gamma'}\,\text{Cut}$$

So in every step the cut is either permuted upwards or becomes less complex. This inevitably pushes the cuts to the top, where they are eliminated. Therefore, every proof can be rewritten without cuts, and this means that Cut is admissible. Recall that in SKSq we just have to consider atomic cuts. Therefore one induction measure, the cut-rank, disappears. There are also semantical proofs for closure under Cut but they are not more efficient. For linear logic there is no semantical proof but the syntactical proof above scales to it.

The main purpose of cut elimination in the sequent calculus is to obtain the following property:

Definition 4.1.3. An inference rule obeys the **subformula property** if every subformula of its premisses occurs in the conclusion. A proof system obeys the property if every rule does.

The subformula property is also called **analyticity**. An immediate consequence is that, when applying an analytic rule bottom-up, there is only a finite number of premisses to choose from. We will call such an inference rule **finitely generating**. This property is desirable from the viewpoint of proof search, since it implies that the search tree is finitely branching. This brings us very close to **decidability**, i.e. all sentences can be proved, or shown non-provable, mechanically.

This is the property we are after when prove the admissibility of Cut, since all the other rules of sequent calculus are analytic. The Cut rule in a sequent system is infinitely generating. Given its conclusion, there is an infinite choice of premisses, corresponding to an infinite choice of cut formulas. Cut-free sequent systems are analytic and thus suitable for theorem proving. In logic programming, proof search corresponds to computation and a proof thus corresponds to a successful execution of a program. Designers of logic programming languages prefer the sequent calculus, because infinite choice and much of the unwanted nondeterminism is limited to the Cut rule, which is admissible.

4.2 Cut Elimination With Splitting

We saw that a normal proof in the sequent calculus is a proof without cuts. In CoS, a normal proof is a proof in the down-fragment:

Definition 4.2.1. A proof in the calculus of structures is **normal** if it does not contain any up-rules.

We have seen that the up-fragment is admissible by the translation to the sequent calculus in Section 2.3. This means that for every proof in CoS there is a proof of the same formula in the down-fragment. Now we will present a method for eliminating the up-fragment purely based on the calculus of structures.

First, we show that all we need is to eliminate the cuts i↑, since the up-fragment can be derived for i↑ and the down-fragment:

Theorem 4.2.1. *Each rule in SKSq is derivable for identity, cut, switch and its dual rule.*

Proof. We replace every instance of $\rho\uparrow\dfrac{S\{T\}}{S\{R\}}$ with the following derivation:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{S\{T\}}{S(T,t)}\,=}{S(T,[R,\bar{R}])}\,\mathrm{i}\downarrow}{S[R,(T,\bar{R})]}\,\mathrm{s}}{S[R,(T,\bar{T})]}\,\rho\downarrow}{S[R,f]}\,\mathrm{i}\uparrow}{S\{R\}}\,=$$

□

So the rules c↑,w↑,u↑ and n↑ are derivable in KSq∪{i↑}. We will not lose anything in provability, even for derivations, if we throw them away. These rules just ensure that the system is symmetric. This is why the up-rules are also called "cuts" and a proof in the down-fragment "cut-free". When we want to normalise a proof we have first to replace any up-rule with the derivation above. Then the only up-rule remaining is the cut rule.

During cut elimination in the sequent calculus, we get into the crucial situation where on one branch a logical rule applies to the main connective of the cut formula and on the other branch the corresponding rule applies to

the dual connective of the dual cut formula. In CoS, rules apply deep inside a context, they are not restricted to main connectives. The methodology of the sequent calculus thus does not apply to the calculus of structures. Instead, we will adopt the technique of splitting, which covers the broadest range of systems in CoS. We will see a cut elimination algorithm for the propositional system SKS, for reasons of simplicity. The method for SKSq will be just sketched, since it follows the same idea.

The first step of the procedure will be the reduction of all cuts to atomic. We are working in SKS so we take that for granted. Before moving on we will need the following definition:

Definition 4.2.2. The following rule is called **super switch**:

$$\mathrm{ss}\downarrow \frac{S\{T\{R\}\}}{S[R, T\{f\}]}$$

Observe that the rule ss↓ takes the formula R from deep inside the context $T\{\ \}$ to a more shallow position. The rule is derivable in SKS:

Lemma 4.2.2. *The rule ss↓ is derivable for* $\{s\}$.

Proof. We will show this by structural induction on $T\{\ \}$. Consider an instance of ss↓:

$$\mathrm{ss}\downarrow \frac{S\{T\{R\}\}}{S[R, T\{f\}]}$$

- If $T\{\ \}$ is the empty context we can replace the given instance by an equivalence rule:

$$= \frac{S\{R\}}{S[R, f]}$$

- If $T\{\ \} = [U, V\{\ \}]$ then, by induction hypothesis, we can apply ss↓ on the context $V\{\ \}$. So we have:

$$\begin{array}{c} \mathrm{ss}\downarrow \dfrac{S[U, V\{R\}]}{S[U, R, V\{f\}]} \\ = \dfrac{}{S[R, [U, V\{f\}]]} \end{array}$$

- If $T\{\ \} = (U, V\{\ \})$ the instance of the rule is derived as follows:

$$\mathrm{ss}\downarrow \frac{S(U, V\{R\})}{\mathrm{s}\ \dfrac{S(U, [R, V\{f\}])}{S[R, (U, V\{f\})]}}$$

We can apply ss↓ on $V\{\ \}$ by induction hypothesis.

□

The atomic cut rule can be replaced by a more restricted version of it. It will then suffice to eliminate this restricted version in order to eliminate all cuts:

Definition 4.2.3. An instance of atomic cut in SKS is called **shallow atomic cut** if it is of the following form:

$$\mathrm{sai}\uparrow \frac{[S, (\alpha, \bar{\alpha})]}{S}$$

In other words, the dual pair of atoms is not inside a context. It is quite simple to reduce all instances of atomic cut to shallow atomic cuts. This will be the second step of the procedure:

Lemma 4.2.3. *The rule ai↑ is derivable for {sai↑,s}.*

Proof. An instance of ai↑ can be replaced by the following derivation:

$$\mathrm{ai}\uparrow \frac{S(\alpha, \bar{\alpha})}{S\{f\}} \qquad \rightsquigarrow \qquad \begin{array}{c} \mathrm{ss}\downarrow \dfrac{S(\alpha, \bar{\alpha})}{[(\alpha, \bar{\alpha}), S\{f\}]} \\ \mathrm{sai}\uparrow \dfrac{}{= \dfrac{[f, S\{f\}]}{S\{f\}}} \end{array}$$

The rule ss↓ can in turn be replaced by a derivation of switches. □

In the sequent calculus there are two proofs above a cut instance. The cut formula is in the conclusion of one proof and the dual of the cut formula is in the conclusion of the other proof. In the calculus of structures we just have one proof above the cut which contains both, the cut formula and its dual. To gain access to two proofs, as in the sequent calculus, we have to use the following lemma:

Lemma 4.2.4. *Each proof* $\begin{array}{c} t \\ \Vert_{KS} \\ T\{\alpha\} \end{array}$ *can be transformed into a proof* $\begin{array}{c} t \\ \Vert_{KS} \\ T\{t\} \end{array}$.

Proof. We replace all occurrences of α by the unit t. Replacements inside the context of any rule instance leave this rule instance intact. Instances of the rules m and s remain intact, also in the case that atom occurrences are replaced inside redex and contractum. Instances of the other rules are replaced by the following derivations:

$$\mathrm{ai}{\downarrow}\,\frac{S\{t\}}{S[\alpha,\bar{\alpha}]} \qquad \rightsquigarrow \qquad \begin{array}{c} = \dfrac{S\{t\}}{S[t,f]} \\ \mathrm{aw}{\downarrow}\,\dfrac{S[t,f]}{S[t,\bar{\alpha}]} \end{array}$$

$$\mathrm{aw}{\downarrow}\,\frac{S\{f\}}{S\{\alpha\}} \qquad \rightsquigarrow \qquad \begin{array}{c} = \dfrac{S\{f\}}{S([t,t],f)} \\ \mathrm{s}\,\dfrac{S([t,t],f)}{S[t,(t,f)]} \\ = \dfrac{S[t,(t,f)]}{S\{t\}} \end{array}$$

$$\mathrm{ac}{\downarrow}\,\frac{S[\alpha,\alpha]}{S\{\alpha\}} \qquad \rightsquigarrow \qquad = \frac{S[t,t]}{S\{t\}}$$

□

Therefore, in the third step of the procedure we choose the topmost cut instance:

$$\begin{array}{c} t \\ \Pi\big\|_{\mathrm{KS}} \\ \mathrm{sai}{\uparrow}\,\dfrac{[R,(\alpha,\bar{\alpha})]}{R} \\ \big\|_{\mathrm{KS}\cup\{\mathrm{sai}\uparrow\}} \\ T \end{array}$$

and we apply the previous lemma twice on the proof Π above it, once for α and once for $\bar{\alpha}$. This way we obtain two different proofs:

$$\begin{array}{cc} t & t \\ \Pi_1\big\|_{\mathrm{KS}} & \Pi_2\big\|_{\mathrm{KS}} \\ [R,\alpha] & [R,\bar{\alpha}] \end{array}$$

This is what we call **splitting**.

It remains one last step. Pick one of the two proofs, let's say Π_1. Replace all occurrences of α by the formula R. Replacements inside the context of any rule instance leave this rule instance intact. Instances of the rules m and s remain intact, also in the case that atom occurrences are replaced inside redex and contractum. Instances of ac↓ and aw↓ are replaced by their general versions:

$$\mathrm{aw}{\downarrow}\,\frac{S\{f\}}{S\{\alpha\}} \qquad \rightsquigarrow \qquad \mathrm{w}{\downarrow}\,\frac{S\{f\}}{S\{R\}}$$

$$\mathrm{ac}{\downarrow}\,\frac{S[\alpha,\alpha]}{S\{\alpha\}} \qquad \rightsquigarrow \qquad \mathrm{c}{\downarrow}\,\frac{S[R,R]}{S\{R\}}$$

These general instances can be easily reduced to atomic instances. The interesting case is the identity. We can't just replace α by R because the inference will not be valid. This is where Π_2 is needed. We will put it in the place of every identity instance that introduces α:

$$\mathrm{ai}{\downarrow}\,\frac{S\{t\}}{S[\alpha,\bar{\alpha}]} \qquad \rightsquigarrow \qquad \begin{array}{c} S\{t\} \\ S\{\Pi_2\}\big\|_{\mathrm{KS}} \\ S[R,\bar{\alpha}] \end{array}$$

This is reminding of the plugging in natural deduction. There we plugged a proof with conlusion A to a premise A of a derivation. Here we have the proof Π_2, which corresponds to the derivation $\begin{array}{c}\alpha\\ \|\\ R\end{array}$, and we plug it to the identity instances of Π_1 that introduce α. This situation becomes available due to the atomicity of i↑.

After the substitution of Π_2 into Π_1 we build the following proof:

$$\begin{array}{c} t \\ \big\|_{\mathrm{KS}} \\ \mathrm{c}{\downarrow}\,\dfrac{[R,R]}{R} \\ \big\|_{\mathrm{KS}\cup\{\mathrm{sai}\uparrow\}} \\ T \end{array}$$

The cut instance has been eliminated. We next choose the new topmost cut and proceed the same way. The procedure described above, also displayed in Figure 4.3, proves the following theorem:

Theorem 4.2.5 (**Cut Elimination**). *Each proof* $\begin{array}{c} t \\ \| \\ T \end{array}_{SKS}$ *can be transformed into a proof* $\begin{array}{c} t \\ \| \\ T \end{array}_{KS}$.

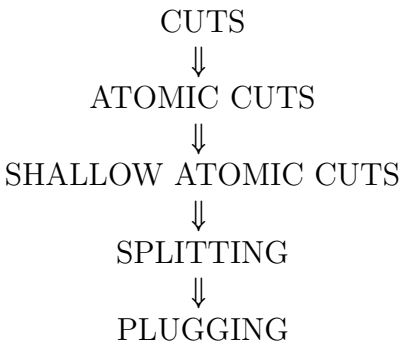


Figure 4.3: Cut Elimination for SKS

Splitting for Predicate Logic

The main difficulty in scaling this procedure to predicate logic is the presence of existential quantifiers in the context of a cut which bind variables in α and $\bar{\alpha}$. This situation prevents the splitting of the proof above the cut because the following equivalence does not hold:

$$\exists x(\alpha(x), \overline{\alpha(x)}) \quad \not\Leftrightarrow \quad \exists x\alpha(x) \text{ and } \exists x\overline{\alpha(x)}$$

The direction from right to left is of course not valid. We will call a cut **splittable** if it is not in the scope of an existential quantifier. The method described above has to be modified in order to deal only with splittable cuts. We will examine in short what changes at each step. The procedure in full detail is in [4].

1. The first thing that we have to do is to turn all cuts to splittable cuts. The following transformation allows us to replace up-rules by splittable cuts:

$$\rho\uparrow\frac{S\{T\}}{S\{R\}} \qquad \rightsquigarrow \qquad \begin{array}{c} =\dfrac{S\{T\}}{(S\{T\},t)} \\ \mathrm{i}\downarrow\dfrac{}{(S\{T\},[S\{R\},\overline{S\{R\}}])} \\ \mathrm{s}\dfrac{}{[S\{R\},(S\{T\},\overline{S\{R\}})]} \\ \rho\downarrow\dfrac{}{[S\{R\},(S\{T\},\overline{S\{T\}})]} \\ \mathrm{i}\uparrow\dfrac{}{=\dfrac{[S\{R\},f]}{S\{R\}}} \end{array}$$

If we consider a context with existential quantifiers as $S\{\ \}$ and $\rho\uparrow$=i$\uparrow$ we see that we can trade an unsplittable cut for a splittable cut with a bigger cut formula. Indeed, think of T as $(R,\bar{R})$. The cut formula is at first R but if we use the derivation above we get a splittable cut with cut formula $S(R,\bar{R})$. Note that we can use this transformation for the reducing of all up-rules instead of that in Theorem 4.2.1. The cuts that will occur will all be splittable.

2. Now, following the method given for SKS, we have to reduce these cuts to atomic, without of course losing splittability. We have seen the reduction of i$\uparrow$ to atomic in section 3.1. It is performed recursively on the complexity of the cut formula. In the cases when the main connective is a conjunction or a disjunction, the procedure reduces the rank of the cut formula replacing a splittable cut by splittable cuts. Let's now remember the following case:

$$\mathrm{i}\uparrow\frac{S(\exists xT,\forall x\bar{T})}{S\{f\}} \qquad \rightsquigarrow \qquad \begin{array}{c} \mathrm{u}\uparrow\dfrac{S(\exists xT,\forall x\bar{T})}{S\{\exists x(T,\bar{T})\}} \\ \mathrm{i}\uparrow\dfrac{}{=\dfrac{S\{\exists xf\}}{S\{f\}}} \end{array}$$

Here, the i$\uparrow$ instance on the left may be splittable but the reduced cut on the right is inside a context $S\{\exists x\{\ \}\}$, and x is free in T (otherwise the given cut would trivially reduce using the vacuous quantifier equation). We can't use this transformation because it destroys splittability. What we do is splitting the cut on the left although it is not atomic. This of course makes its plugging a more complex issue.

3. Next we should replace the atomic cuts by shallow atomic cuts and split the topmost among them. But the sai$\uparrow$ for predicate calculus is the following:

$$\text{sai}\uparrow \frac{[S, \exists x(\alpha, \bar{\alpha})]}{S}$$

where $\exists x$ denotes a sequence of quantifiers that existentially close $(a, \bar{a})$. This is not a splittable cut since it is in the scope of those existential quantifiers. We can not reduce cuts to shallow and keep them splittable at the same time. We will have to do the splitting first, and then we obtain shallowness by applying ss$\downarrow$ to each part separately.

4. The plugging of the two proofs happens as in the propositional case. Then we proceed to the next instance of ai$\uparrow$.

One could ask why to bother and try for internal methods when we can eliminate cuts through translations to the sequent calculus. That is an algorithmic procedure as well. After all, we usually just need a proof of cut admissibility in order to have analyticity. However, there are several reasons to study techniques for normalisation purely based on the calculus of structures:

- Calculus of structures is no more depended on sequent calculus and becomes an autonomous formalism for logical reasoning.
- We could get a simpler algorithm for eliminating cuts compared to that of sequent calculus and thus take some interesting results about the complexity of cut elimination procedures.
- There are several kinds of logic, such as intermediate, substructural and modal logics, that there is not a cut-free sequent calculus for them. This explains why some of these logics are usually formalised using **hypersequents**:

  $$\Gamma_1 \vdash \Delta_1 \quad | \quad \Gamma_2 \vdash \Delta_2 \quad | \quad \cdots \quad | \quad \Gamma_n \vdash \Delta_n$$

  The structural connective | is interpreted as a disjunction. So a hypersequent is a disjunction of ordinary sequents. In this extended formalism a cut-free system may exist for a logic that does not have a cut-free sequent system. A syntactic cut elimination for SKS could be useful in order to obtain cut-free systems for these logics in CoS without having to translate the complicated structure of hypersequents.
- SKS admits a cut elimination procedure which is similar to normalisation in natural deduction, i.e. plugging proofs to the premisses. This could lead to a computational interpretation of the proof system. Recall that the plugging corresponds to passing an input to a function in the λ-calculus.

4.3 Normalisation With Decomposition

In this section we are going to generalise the notion of normalisation for arbitrary derivations. As we have seen, the normal form of a proof of R in SKSgq is a proof of R in KSgq, that is the proof obtained after the elimination of the up-fragment. But KSgq is not strongly equivalent to SKSgq. In other words, we can not eliminate the up-rules from an arbitrary derivation. So we have to define what is a normal form for a derivation. The following definition is not characterised by the absence of certain inference rules, but by the the way in which the inference rules are composed:

Definition 4.3.1. We will call a derivation $\begin{array}{c} A \\ \| \\ B \end{array}_{\mathrm{SKSgq}}$ **normal** if there is no up-rule below a down-rule.

So in a normal derivation the up-rules are applied first and then follows a part with only down-rules. We will call this separation **decomposition**. A normal derivation has the following shape:

$$\begin{array}{c} A \\ \|_{\uparrow} \\ C \\ \|_{\downarrow} \\ B \end{array}$$

This is similar to what happens in the sequent calculus. During cut elimination, instances of Cut are pushed to the top of the proof. There, the cuts interact with the Axiom instances and vanish. But if we have to do with derivations instead of proofs this can not happen. The cuts do not interact with arbitrary non-logical axioms. So we obtain a derivation where cuts are not eliminated but are moved to the top, right below the axioms. There is also a similarity with natural deduction. A normal proof in Nc consists of an upper part with elimination rules and a down part with only introduction rules.

The definition subsumes the definition of a normal proof (Definition 4.2.1). A normal proof is of course a normal derivation, just without the up-fragment part. For the opposite consider a proof, which is a derivation with premise syntactically equivalent to t. Suppose that we get its normal form, if seen as a derivation:

$$\begin{array}{c} t \\ \Big\Vert\uparrow \\ P \\ \Big\Vert\downarrow \\ R \end{array}$$

Since the conclusion of all rules in the up-fragment is equivalent to t if their premise is equivalent to t, then P has to be equivalent to t. We thus have a proof of R in the down-fragment, a normal proof.

The question is if, given a derivation, there is always a derivation in normal form with the same premise and conclusion. Brünnler showed this by a translation to sequent calculus in [5]. The system he used is a variant of G1c, which is restricted to formulas in negation normal form (Figure 4.4). This explains the two additional axiom rules ($\vdash A, \bar{A}$, $A, \bar{A} \vdash$) and the absence of the rules for implication $\rightarrow_L$ and $\rightarrow_R$. Let us call this system LK, as the original Gentzen's system, although there are slight differences. The proof is resembling of that of equivalence between GS1 and SKSgq. We will use again the translations $\underline{\cdot}_G$ and $\underline{\cdot}_S$.

Let us first examine the direction from CoS to sequent calculus:

Lemma 4.3.1. *For each derivation from A to B in SKSgq there is a proof of $\underline{A}_G \vdash \underline{B}_G$ in LK + Cut.*

Proof. We will prove the theorem by induction on the length of the given derivation in SKSgq. In proofs we will drop the subscript of the translation to improve readability:

- If the derivation consists of just one formula B then the corresponding derivation in GS1 consists of just one sequent $B \vdash B$.
- We single out the topmost rule instance of the derivation Δ in CoS:

$$\begin{array}{c} A \\ \Delta \Big\Vert \text{SKSgq} \\ B \end{array} \quad = \quad \rho\, \dfrac{S\{T\}}{S\{R\}} \quad \begin{array}{c} \\ \Delta' \Big\Vert \text{SKSgq} \\ B \end{array}$$

$$\mathrm{Ax}\ \frac{}{A \vdash A} \qquad \top\ \frac{}{\vdash \top}$$

$$\mathrm{Ax}\ \frac{}{\vdash A, \bar{A}} \qquad \mathrm{Ax}\ \frac{}{A, \bar{A} \vdash}$$

$$\mathrm{c}_L\ \frac{\Phi, A, A \vdash \Psi}{\Phi, A \vdash \Psi} \qquad \mathrm{c}_R\ \frac{\Phi \vdash A, A, \Psi}{\Phi \vdash A, \Psi}$$

$$\mathrm{w}_L\ \frac{\Phi \vdash \Psi}{\Phi, A \vdash \Psi} \qquad \mathrm{w}_R\ \frac{\Phi \vdash \Psi}{\Phi \vdash A, \Psi}$$

$$\wedge_L\ \frac{\Phi, A, B \vdash \Psi}{\Phi, A \wedge B \vdash \Psi} \qquad \wedge_R\ \frac{\Phi \vdash A, \Psi \qquad \Phi' \vdash B, \Psi'}{\Phi, \Phi' \vdash A \wedge B, \Psi, \Psi'}$$

$$\vee_L\ \frac{A, \Phi \vdash \Psi \qquad B, \Phi' \vdash \Psi'}{\Phi, \Phi', A \vee B \vdash \Psi, \Psi'} \qquad \vee_R\ \frac{\Phi \vdash A, B, \Psi}{\Phi \vdash A \vee B, \Psi}$$

$$\forall_L\ \frac{\Phi, A[x/t] \vdash \Psi}{\Phi, \forall x A \vdash \Psi} \qquad \forall_R\ \frac{\Phi \vdash A[x/y], \Psi}{\Phi \vdash \forall x A, \Psi}\ y \notin FV\{\Phi, \forall x A, \Psi\}$$

$$\exists_L\ \frac{\Phi, A[x/y] \vdash \Psi}{\Phi, \exists x A \vdash \Psi}\ y \notin FV\{\Phi, \exists x A, \Psi\} \qquad \exists_R\ \frac{\Phi \vdash A[x/\tau], \Psi}{\Phi \vdash \exists x A, \Psi}$$

Figure 4.4: System LK

The corresponding derivation in GS1 will be as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{}\,\Pi}{T \vdash R}\qquad \dfrac{\dfrac{}{}\,\Delta_1}{S\{T\} \vdash S\{R\}} \qquad \dfrac{\dfrac{}{}\,\Delta_2}{S\{R\} \vdash B}}{S\{T\} \vdash B}\,\mathrm{Cut}}{}$$

where Δ_1 exists by a slight modification of Lemma 2.2.1 and Δ_2 exists by induction hypothesis. The proof Π depends on the rule ρ. Thus it will be enough to show that for every rule $\rho\,\dfrac{S\{T\}}{S\{R\}}$ of SKSgq there exists a proof $\dfrac{\dfrac{}{}\,\Pi}{T \vdash R}$ in GS1:

$$\mathrm{i}{\downarrow}\,\frac{S\{t\}}{S[R, \bar{R}]} \qquad \rightsquigarrow \qquad \dfrac{}{\vdash R, \bar{R}}\,\mathrm{Ax}$$

$$\mathrm{i}{\uparrow}\,\frac{S(R, \bar{R})}{S\{f\}} \qquad \rightsquigarrow \qquad \dfrac{}{R, \bar{R} \vdash}\,\mathrm{Ax}$$

$$\mathrm{s}\,\frac{S([R, U], T)}{S[(R, T), U]} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{R \vdash R}\,\mathrm{Ax} \quad \dfrac{}{U \vdash U}\,\mathrm{Ax}}{R \vee U \vdash R, U}\,\vee_L \qquad \dfrac{}{T \vdash T}\,\mathrm{Ax}}{T, R \vee U \vdash R \wedge T, U}\,\wedge_R}{T \wedge (R \vee U) \vdash R \wedge T, U}\,\wedge_L}{T \wedge (R \vee U) \vdash (R \wedge T) \vee U}\,\vee_R$$

$$\mathrm{w}{\downarrow}\,\frac{S\{f\}}{S\{R\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{}{\vdash \top}\,\top}{\vdash R, \top}\,\mathrm{w}_R$$

$$\mathrm{w}{\uparrow}\,\frac{S\{R\}}{S\{t\}} \qquad \rightsquigarrow \qquad \mathrm{w}_L\,\frac{\top\,\dfrac{}{\vdash \top}}{R \vdash \top}$$

$$\mathrm{c}{\downarrow}\,\frac{S[R,R]}{S\{R\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\mathrm{Ax}\,\dfrac{}{R \vdash R} \qquad \mathrm{Ax}\,\dfrac{}{R \vdash R}}{\vee_L\,\dfrac{R \vee R \vdash R, R}{\mathrm{c}_R\,\dfrac{R \vee R \vdash R}{}}}}{}$$

$$\mathrm{c}{\uparrow}\,\frac{S\{R\}}{S(R,R)} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\mathrm{Ax}\,\dfrac{}{R \vdash R} \qquad \mathrm{Ax}\,\dfrac{}{R \vdash R}}{\wedge_R\,\dfrac{R, R \vdash R \wedge R}{\mathrm{c}_L\,\dfrac{R \vdash R \wedge R}{}}}}{}$$

$$\mathrm{u}{\downarrow}\,\frac{S\{\forall x[R,T]\}}{S[\forall x R, \exists x T]} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{Ax}\,\dfrac{}{R \vdash R} \qquad \mathrm{Ax}\,\dfrac{}{T \vdash T}}{\vee_L\,\dfrac{R \vee T \vdash R, T}{}}}{\exists_R\,\dfrac{R \vee T \vdash R, \exists x T}{}}}{\forall_L\,\dfrac{\forall x(R \vee T) \vdash R, \exists x T}{}}}{\forall_R\,\dfrac{\forall x(R \vee T) \vdash \forall x R, \exists x T}{}}}{\vee_R\,\dfrac{\forall x(R \vee T) \vdash \forall x R \vee \exists x T}{}}$$

$$\mathrm{u}{\uparrow}\,\frac{S(\exists x R, \forall x T)}{S\{\exists x(R,T)\}} \qquad \rightsquigarrow \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{Ax}\,\dfrac{}{R \vdash R} \qquad \mathrm{Ax}\,\dfrac{}{T \vdash T}}{\wedge_R\,\dfrac{R, T \vdash R \wedge T}{}}}{\exists_R\,\dfrac{R, T \vdash \exists x(R \wedge T)}{}}}{\forall_L\,\dfrac{R, \forall T \vdash \exists x(R \wedge T)}{}}}{\exists_L\,\dfrac{\exists x R, \forall x T \vdash \exists x(R \wedge T)}{}}}{\wedge_L\,\dfrac{\exists x R \wedge \forall x T \vdash \exists x(R \wedge T)}{}}$$

$$\mathrm{n}{\downarrow}\,\frac{S\{R[x/\tau]\}}{S\{\exists x R\}} \quad \rightsquigarrow \quad \exists_R\,\frac{\mathrm{Ax}\,\dfrac{}{R[x/\tau] \vdash R[x/\tau]}}{R[x/\tau] \vdash \exists x R}$$

$$\mathrm{n}{\uparrow}\,\frac{S\{\forall x R\}}{S\{R[x/\tau]\}} \quad \rightsquigarrow \quad \forall_L\,\frac{\mathrm{Ax}\,\dfrac{}{R[x/\tau] \vdash R[x/\tau]}}{\forall x R \vdash R[x/\tau]}$$

□

For the direction from LK to SKSgq we can show the following:

Lemma 4.3.2. *For each proof of $\Phi \vdash \Psi$ in LK there is a derivation in normal form from $\underline{\Phi}_S$ to $\underline{\Psi}_S$ in SKSgq.*

Proof. By induction on the depth of the proof tree:

- The base cases are:

$$\mathrm{Ax}\,\frac{}{\vdash A, \bar{A}} \quad \rightsquigarrow \quad \mathrm{i}{\downarrow}\,\frac{t}{[A, \bar{A}]}$$

$$\mathrm{Ax}\,\frac{}{A, \bar{A} \vdash} \quad \rightsquigarrow \quad \mathrm{i}{\uparrow}\,\frac{(A, \bar{A})}{f}$$

$$\mathrm{Ax}\,\frac{}{A \vdash A} \quad \rightsquigarrow \quad \mathrm{A}$$

- Now we will consider proofs in LK assuming that the part above the last rule can be translated in normal form (induction hypothesis). We have to check all the possible last rule instances:

$$\mathrm{c}_L\,\frac{\Phi, A, A \vdash \Psi}{\Phi, A \vdash \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \mathrm{c}\uparrow\,\dfrac{(\Phi, A)}{(\Phi, A, A)} \\ \Big\|\uparrow \\ C \\ \Big\|\downarrow \\ \Psi \end{array}$$

$$\mathrm{c}_R\,\frac{\Phi \vdash A, A, \Psi}{\Phi \vdash A, \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \Phi \\ \Big\|\uparrow \\ C \\ \Big\|\downarrow \\ \mathrm{c}\downarrow\,\dfrac{[A, A, \Psi]}{[A, \Psi]} \end{array}$$

$$\mathrm{w}_L\,\frac{\Phi \vdash \Psi}{\Phi, A \vdash \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \mathrm{w}\uparrow\,\dfrac{(\Phi, A)}{\Phi} \\ \Big\|\uparrow \\ C \\ \Big\|\downarrow \\ \Psi \end{array}$$

$$\mathrm{w}_R\,\frac{\Phi \vdash \Psi}{\Phi \vdash A, \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \Phi \\ \Big\|\uparrow \\ C \\ \Big\|\downarrow \\ \mathrm{w}\downarrow\,\dfrac{\Psi}{[A, \Psi]} \end{array}$$

$$\wedge_L \frac{\Phi, A, B \vdash \Psi}{\Phi, A \wedge B \vdash \Psi} \qquad \rightsquigarrow \qquad \begin{array}{c} = \dfrac{(\Phi, (A, B))}{(\Phi, A, B)} \\ \Big\Vert\uparrow \\ C \\ \Big\Vert\downarrow \\ \Psi \end{array}$$

In the two following cases we get two normal derivations by induction hypothesis, and they have to be taken apart and composed in the right way to yield the normal derivation:

$$\wedge_R \frac{\Phi \vdash A, \Psi \qquad \Phi' \vdash B, \Psi'}{\Phi, \Phi' \vdash A \wedge B, \Psi, \Psi'} \qquad \rightsquigarrow \qquad \begin{array}{c} (\Phi \ , \ \Phi') \\ \Big\Vert\uparrow \quad \Big\Vert\uparrow \\ (C \ , \ C') \\ \Big\Vert\downarrow \quad \Big\Vert\downarrow \\ \mathrm{s}\dfrac{([A, \Psi], [B, \Psi'])}{\mathrm{s}\dfrac{[([A, \Psi], B), \Psi']}{[(A, B), \Psi, \Psi']}} \end{array}$$

$$\vee_L \frac{A, \Phi \vdash \Psi \qquad B, \Phi' \vdash \Psi'}{\Phi, \Phi', A \vee B \vdash \Psi, \Psi'} \qquad \rightsquigarrow \qquad \begin{array}{c} \mathrm{s}\dfrac{([A, B], \Phi, \Phi')}{\mathrm{s}\dfrac{([(A, \Phi), B], \Phi')}{[(A, \Phi), (B, \Phi')]}} \\ \Big\Vert\uparrow \quad \Big\Vert\uparrow \\ [C \ , \ C'] \\ \Big\Vert\downarrow \quad \Big\Vert\downarrow \\ [\Psi \ , \ \Psi'] \end{array}$$

$$\vee_R \frac{\Phi \vdash A, B, \Psi}{\Phi \vdash A \vee B, \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \Phi \\ \Big\Vert\uparrow \\ C \\ \Big\Vert\downarrow \\ =\dfrac{[A,B,\Psi]}{[[A,B],\Psi]} \end{array}$$

$$\forall_L \frac{\Phi, A[x/t] \vdash \Psi}{\Phi, \forall x A \vdash \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \mathrm{n}\uparrow\dfrac{(\Phi, \forall x A)}{(\Phi, A[x/t])} \\ \Big\Vert\uparrow \\ C \\ \Big\Vert\downarrow \\ \Psi \end{array}$$

$$\forall_R \frac{\Phi \vdash A[x/y], \Psi}{\Phi \vdash \forall x A, \Psi}\, y \notin FV\{\Phi, \forall x A, \Psi\} \quad \rightsquigarrow \quad \begin{array}{c} =\dfrac{\Phi}{\forall y \Phi} \\ \Big\Vert\uparrow \\ \forall y C \\ \Big\Vert\downarrow \\ \mathrm{u}\downarrow\dfrac{\forall y[A[x/y], \Psi]}{=\dfrac{[\forall y A[x/y], \forall y \Psi]}{=\dfrac{[\forall x A, \forall y \Psi]}{[\forall x A, \Psi]}}} \end{array}$$

In the derivation above we used three times the equivalence relation, once for variable renaming and twice for vacuous quantifier. Those equivalences carry a proviso. In the instances above, y should not be free in Φ, Ψ or A. The proviso on the eigenvariable of $\forall_R$ is exactly what is needed to ensure those conditions. The same situation occurs in the following case:

$$\exists_L \frac{\Phi, A[x/y] \vdash \Psi}{\Phi, \exists x A \vdash \Psi}\, y \notin FV\{\Phi, \exists x A, \Psi\} \quad \rightsquigarrow \quad \begin{array}{c} = \dfrac{(\Phi, \exists x A)}{(\Phi, \exists y A[x/y])} \\ = \dfrac{(\Phi, \exists y A[x/y])}{(\forall y \Phi, \exists y A[x/y])} \\ \mathrm{u}\uparrow \dfrac{(\forall y \Phi, \exists y A[x/y])}{\exists y (\Phi, A[x/y])} \\ \Big\|\uparrow \\ C \\ \Big\|\downarrow \\ = \dfrac{\exists y \Psi}{\Psi} \end{array}$$

$$\exists_R \frac{\Phi \vdash A[x/\tau], \Psi}{\Phi \vdash \exists x A, \Psi} \quad \rightsquigarrow \quad \begin{array}{c} \Phi \\ \Big\|\uparrow \\ C \\ \Big\|\downarrow \\ \mathrm{n}\downarrow \dfrac{[A[x/t], \Psi]}{[\exists x A, \Psi]} \end{array}$$

Observe that in all cases we added either an up-rule to the top or a down-rule to the bottom so that the derivation remains in normal form. □

Note that the translation of Cut does not yield a normal derivation:

$$\mathrm{Cut}\,\frac{\Phi \vdash A, \Psi \qquad \Phi', A \vdash \Psi'}{\Phi, \Phi' \vdash \Psi, \Psi'} \quad \rightsquigarrow \quad \begin{array}{c} (\Phi' \ , \ \Phi) \\ \Big\|{\uparrow} \\ (\Phi' \ , \ C) \\ \Big\|{\downarrow} \\ \mathrm{s}\,\dfrac{(\Phi', [A, \Psi])}{[\Psi, (\Phi', A)]} \\ \Big\|{\uparrow} \\ [\Psi \ , \ C'] \\ \Big\|{\downarrow} \\ [\Psi \ , \ \Psi'] \end{array}$$

This is natural since translations between different formalisms should always respect the notion of normal form. We also saw that in the translation of Section 2.2, where only proofs are involved, Cut translates to the corresponding redex i↑ and vice versa. For another example, when we translate from natural deduction to sequent calculus, normal derivations correspond to cut-free proofs.

In the translation above we can notice a connection between the rules of the sequent calculus and those of the calculus of structures. Each left rule is translated by adding an up-rule to the top of the derivation. The right rules correspond to down-rules added to the bottom. If we look at the proof of the equivalence between natural deduction and sequent systems (see [1]) we will notice an interesting similarity. Suppose that we translate inductively the proof of a sequent $\Phi \vdash A$ into a deduction with conclusion A and open assumptions a subset of Φ. If the last rule is a right rule then we translate it by adding an introduction rule to the bottom of the deduction. Left rules correspond to elimination rules added to the top. So we could say that in all three formalisms rules come in dual pairs, and there is a connection between those kinds of duality (Figure 4.5). We could also make a straightforward connection between CoS and Nc by recalling that a normal derivation in natural deduction has an upper part with elimination rules and a down part with introduction rules.

Natural Deduction	Sequent Calculus	Calculus of Structures
Introduction Rules	Right Rules	Down-Rules
Elimination Rules	Left Rules	Up-Rules

Figure 4.5: Dualities

The two lemmas above lead us to the following theorem:

Theorem 4.3.3 (**Normalisation**). *For every derivation* $\begin{array}{c} A \\ \| \\ B \end{array}$ *there is a derivation* $\begin{array}{c} A \\ \| \\ C \\ \| \\ B \end{array}$.

Proof. We just follow the steps (Figure 4.6):

1. We translate the given derivation into a proof of the sequent $A \vdash B$ (Lemma 4.3.1).
2. We eliminate cuts and obtain a cut-free proof.
3. We translate the cut-free proof of $A \vdash B$ into a derivation from A to B (Lemma 4.3.2). The translation guarantees that the derivation is in normal form.

The reason for cut elimination (step 2) is that the Cut rule can not be translated into a normal derivation. Therefore, Lemma 4.3.2 does not hold for GS1 + Cut. □

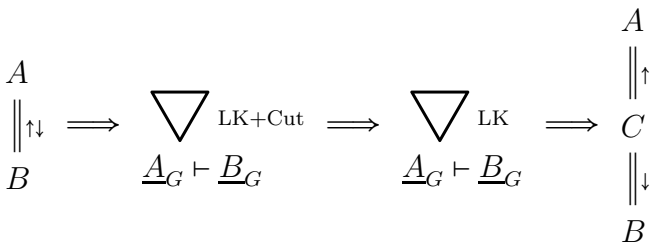


Figure 4.6: Normalisation Through Translation

This decomposition theorem can be seen as the symmetric closure of cut elimination. The normal form of a derivation is a generalisation of the normal form of a proof. Therefore, we obtain as a corollary what we have already proved by translation (Theorem 2.3.1) and syntactically (Theorem 4.2.5):

Corollary 4.3.4. *For every proof* $\begin{matrix} t \\ \| \\ A \end{matrix}\!\uparrow\downarrow$ *there is a proof* $\begin{matrix} t \\ \| \\ A \end{matrix}\!\downarrow$.

It is an interesting question whether we can prove the normalisation theorem without the detour via the sequent calculus. The point of proving internally the same theorem is that the method could be applied to logics which do not have a cut-free sequent calculus. An independent procedure of normalisation could even lead to an approach to computation. A normal derivation $\begin{matrix} Q \\ \| \\ R \end{matrix}$ could be seen as a function $f : Q \to R$ and a possible input u of type Q could be represented by the normal derivation $\begin{matrix} P \\ \| \\ Q \end{matrix}$. Their composition:

$$\begin{matrix} P \\ \|\uparrow \\ C \\ \|\downarrow \\ Q \\ \|\uparrow \\ C' \\ \|\downarrow \\ R \end{matrix}$$

is not of course in normal form. This normalisation would correspond to the computation of $f(u)$.

An internal method should be based on permutations of rules:

Definition 4.3.2. We say that a rule ρ **permutes over** rule π, or rule π **permutes under** ρ, if for every derivation $\pi\dfrac{\dfrac{T}{U}}{R}\rho$ there is a derivation $\rho\dfrac{\dfrac{T}{V}}{R}\pi$ for some formula V.

Permuting an instance of a rule over an instance of another rule in a derivation means exchanging the two rules without breaking the derivation. While it is easy to come up with local proof transformations that normalise a derivation if they terminate, the presence of contraction makes termination hard to show. This has been achieved for some systems related to linear logic, by essential use of the fact that contraction is restricted. However, a syntactic normalisation procedure for derivations in SKSgq remains an open problem.

Interpolation

In system SKSgq there are rules that, when going up in a derivation, introduce new predicate symbols: the cut rule and the co-weakening rule. Dually, the identity and the weakening rule are the only rules that introduce new predicate symbols when going down. Using the normalisation theorem 4.3.3, a derivation is separated into two phases. In the top one, we have rules that do not introduce new atoms going down, since there are only up-rules and therefore no identity or weakening. In the bottom phase, rules do not introduce new atoms going up because there is no cut or co-weakening instance. Consequently, the formula in between contains only atoms that occur both in the premise and in the conclusion of the derivation. This proves the following theorem:

Theorem 4.3.5 (**Craig Interpolation**)**.** *For all formulas P and Q, if P implies Q then there is a formula V such that P implies V, V implies Q and all the predicate symbols that occur in V occur in both P and Q.*

The formula V is called an **interpolant**. William Craig proved this theorem in 1957 by model theoretic means. Thus, the formula that connects the up-fragment with the down-fragment of a derivation is an interpolant.

In sequent calculus, interpolation can be proved via cut elimination. The subformula property holds, which makes Craig interpolation provable by an induction over the derivations. In CoS we have a similar situation: both cut elimination and interpolation are immediate consequences of a derivation's normal form.

4.4 Normalisation Without Cut Elimination

We have already mentioned that the motivation for cut elimination in the sequent calculus is to obtain the subformula property (analyticity). A cut-free system is a finitely generating system, which means that when we apply a rule bottom-up there is only a finite number of premisses to choose from. Here we will present a finitely generating system in the calculus of structures that does not presuppose the elimination of all cuts. We will describe the system for the propositional logic. So, from now on, α is a propositional variable and not an atomic type. However, everything that follows scales to predicate logic as well.

There are two infinitely generating rules in system SKS: the atomic co-weakening and the atomic cut rule. The rule $\mathrm{aw}\uparrow \dfrac{S\{\alpha\}}{S\{t\}}$ is clearly infinitely generating since there is an infinite choice of atoms α, but it can be eliminated by deriving it for cut, switch and weakening (see Theorem 4.2.1):

$$\mathrm{aw}\uparrow \frac{S\{\alpha\}}{S\{t\}} \qquad \rightsquigarrow \qquad \begin{array}{r} = \dfrac{S\{\alpha\}}{S(\alpha,t)} \\ = \dfrac{S(\alpha,[f,t])}{S[t,(\alpha,f)]} \\ \mathrm{s}\dfrac{S[t,(\alpha,f)]}{S[t,(\alpha,\bar{\alpha})]} \\ \mathrm{aw}\downarrow \dfrac{S[t,(\alpha,\bar{\alpha})]}{S[t,f]} \\ \mathrm{ai}\uparrow \dfrac{S[t,f]}{S\{t\}} \\ = \end{array}$$

Thus the infinity of aw↑ can be reduced to the infinity of ai↑.

There are two sources of infinite choice in the Cut rule of sequent calculus: an infinite choice of atoms and an infinite choice in how these atoms can be combined for making the cut formula. But in the calculus of structures one can reduce the cut rule i↑ to its atomic form ai↑. Therefore, the second source of infinity can be ignored. The only factor of infinity is the choice of the atom that ai↑ will introduce bottom-up.

Let us now introduce a variant of ai↑:

Definition 4.4.1. The following rule is called **finitely generating atomic cut**:

$$\mathrm{fai}\uparrow \frac{S(\alpha,\bar{\alpha})}{S\{f\}} \qquad \text{where } \alpha \text{ or } \bar{\alpha} \text{ appears in the conclusion}$$

This is an analytic cut rule. It only introduces atoms that occur in the conclusion, which implies a choice over a finite set.

So we can discriminate between two kinds of cut when they are atomic: those that introduce new atoms going up and those that they don't. If we are interested just in analyticity, all we need to do is to eliminate the first kind. Let's see how we can do this. Take a proof of R in the propositional system SKS:

1. First, replace every instance of aw↑ with the equivalent derivation shown above. Thus we obtain a proof of R in SKS$\setminus\{\text{aw}\uparrow\}$.
2. Single out the bottommost instance of ai↑ that violates the proviso of fai↑:

$$\begin{array}{c} t \\ \| \\ {\scriptstyle \text{SKS}\setminus\{\text{aw}\uparrow\}} \\ \text{ai}\uparrow \dfrac{S(\alpha,\bar{\alpha})}{S\{f\}} \\ \| \\ {\scriptstyle (\text{SKS}\setminus\{\text{ai}\uparrow,\text{aw}\uparrow\})\cup\{\text{fai}\uparrow\}} \\ R \end{array}$$

3. Replace all instances of α and $\bar{\alpha}$ in the proof above the cut with t and f respectively. We will check that all rule instances still hold or become instances of the equivalence rule:

$$\text{ai}\downarrow \frac{S\{t\}}{S[\alpha,\bar{\alpha}]} \quad \rightsquigarrow \quad = \frac{S\{t\}}{S[t,f]}$$

$$\text{ai}\uparrow \frac{S(\alpha,\bar{\alpha})}{S\{f\}} \quad \rightsquigarrow \quad = \frac{S(t,f)}{S\{f\}}$$

$$\text{ac}\downarrow \frac{S[\alpha,\alpha]}{S\{\alpha\}} \quad \rightsquigarrow \quad = \frac{S[t,t]}{S\{t\}}$$

$$\text{ac}\uparrow \frac{S\{\alpha\}}{S(\alpha,\alpha)} \quad \rightsquigarrow \quad = \frac{S\{t\}}{S(t,t)}$$

$$\text{aw}\downarrow \frac{S\{f\}}{S\{\alpha\}} \quad \rightsquigarrow \quad \text{aw}\downarrow \frac{S\{f\}}{S\{t\}}$$

Instances of m and s remain of course intact.

4. Proceeding inductively upwards, remove all infinitary atomic cuts.

This is a weak form of cut elimination. Restricting the generic cut rule to fai↑ will only entail a linear cost for the size of proofs. The full cut elimination returns exponentially bigger proofs. Thus, if we want a finitely generating system without increasing too much the size of the proofs, we should just restrict the use of ai↑ to atoms appearing in the conclusion. Note that we can't extend the method described above to a full cut elimination. If we apply the method to a cut instance such that α or $\bar{\alpha}$ appears in $S\{\ \}$, this could destroy the derivation from $S\{f\}$ to R.

We now define the finitely generating system FKS to be:

$$(\mathrm{SKS} \setminus \{\mathrm{ai}\uparrow, \mathrm{aw}\uparrow\}) \cup \{\mathrm{fai}\uparrow\}$$

We can see the system in Figure 4.7. The following theorem has been already justified:

Theorem 4.4.1. *Each formula is provable in system SKS if and only if it is provable in system FKS.*

Proof. Given a proof in SKS, we can derive aw↑ for ai↓,ai↑,aw↓ and s. Then we can eliminate every instance of ai↑ that is not an instance of fai↑ using the algorithm described above. □

In system FKS no rule, seen bottom-up, introduces new atoms. It thus satisfies the main aspect of the subformula property: when given a conclusion of a rule there is only a finite number of premisses to choose from. All its rules can only be applied in a finite number of different ways. Therefore, in proof search the branching of the search tree is finite. The only infinity that remains is in the unboundedness of the proofs themselves. All this can be also achieved in SKS and in sequent calculus through cut elimination. The point here is that it is possible to eliminate infinite choice without having to deal with exponentially bigger proofs.

As far as predicate logic is concerned, there is another source of infinite choice added to aw↑ and atomic cut. We refer to the choice in instantiating an existentially quantified variable. Recall the rule:

$$\mathrm{n}\downarrow \frac{S\{R[x/\tau]\}}{S\{\exists x R\}}$$

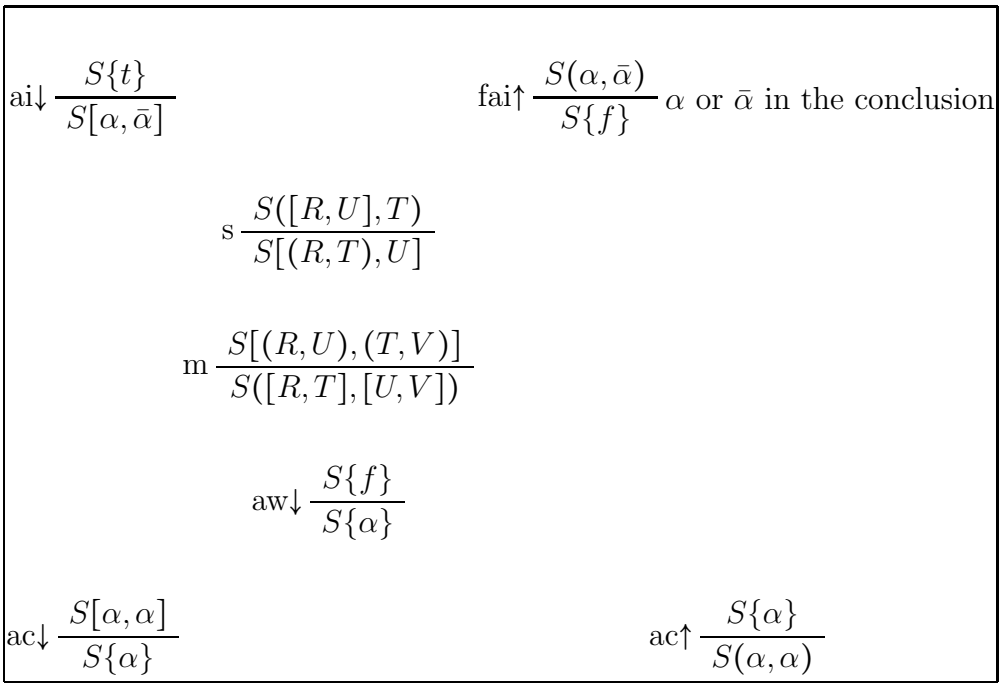


Figure 4.7: System FKS

If you look at the rule bottom-up there is an infinite choice for the term t, it could be any term of the language. Remember that terms are defined recursively from constants, variables and functions. Rule n↓ can be reduced to an instantiation rule which introduces one variable and an instantiation rule that introduces one function symbol or a fixed constant, all chosen among those that occur in the conclusion. These of course will be finitely generating instantiation rules. Details in [3].

In the sequent calculus the Cut rule is considered as the use of a lemma in a proof. Indeed, if we look carefully at the following instance of the two-sided version:

$$\text{Cut}\,\frac{\vdash A \qquad A \vdash B}{\vdash B}$$

we can imagine the left premise as the proof of a lemma A and the right as the proof of B given A. So when we eliminate cuts we create a direct, analytic proof with no lemmas. In CoS the i↑ rule can be seen the same way. The rate of growth of a proof under cut elimination is hyperexponential. This shows the importance of indirect reasoning in mathematics. Without the use of lemmas we cannot present proofs of manageable size.

Alessio Guglielmi suggests in [6] that when the cut formula appears in the conclusion then the cut rule corresponds not to the use of a lemma but to a case analysis. If we eliminate it then the proof will be essentially different. Let's see an example. Consider the following valid formula:

$$F = \exists x \forall y (p(x) \to p(y))$$

It is known as **the drinker property**. We can prove it by knowing the two lemmas:

L_1 : If $\forall z\, p(z)$ then F is true, because the conclusion $p(y)$ is always true. In sequent terms this is written $\forall z\, p(z) \vdash F$. Its equivalent one-sided form is $\vdash \neg \forall z\, p(z), F$.

L_2 : If $\neg \forall z\, p(z)$ then F is true, because the premise $p(x)$ can be falsified. In sequent style this is written $\neg \forall z\, p(z) \vdash F$.

Assume that Π_1 and Π_2 are the formal proofs of L_1 and L_2 respectively. Then we can build the following proof in the sequent calculus:

$$\dfrac{\dfrac{\begin{matrix}\Pi_1 \\ \vdash \neg \forall z\, p(z), F\end{matrix} \qquad \begin{matrix}\Pi_2 \\ \neg \forall z\, p(z) \vdash F\end{matrix}}{\vdash F, F}\,\mathrm{Cut}}{\vdash F}\,\mathrm{c}_R$$

The two lemmas are no more than the case analysis about the truth of $\forall z\, p(z)$. If we eliminate the cut from the proof above we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p(x), p(z) \vdash p(z), p(y)}\,\mathrm{Ax}}{p(z) \vdash p(x) \to p(z), p(y)}\,{\to_R}}{\vdash p(x) \to p(z), p(z) \to p(y)}\,{\to_R}}{\vdash p(x) \to p(z), \forall y (p(z) \to p(y))}\,{\forall_R}}{\vdash p(x) \to p(z), \exists x \forall y (p(x) \to p(y))}\,{\exists_R}}{\vdash \forall y (p(x) \to p(y)), \exists x \forall y (p(x) \to p(y))}\,{\forall_R}}{\vdash \exists x \forall y (p(x) \to p(y)), \exists x \forall y (p(x) \to p(y))}\,{\exists_R}}{\vdash \exists x \forall y (p(x) \to p(y))}\,\mathrm{c}_R$$

According to Guglielmi, this proof is intuitively different than the other one corresponding to the case analysis. It is not just the direct version. Thus, if we want the normal form of a proof to contain its essence, maybe we shouldn't eliminate all cuts.

Consistency

An application of the finitely generating system FKS is the proof of the syntactical consistency of SKS without going through cut elimination. As we mentioned before, the main advantage of finite choice is that we can show that a formula is non-provable:

Theorem 4.4.2. *The unit f is not provable in system FKS.*

Proof. We have to show that there is no proof:

$$\begin{array}{c} t \\ \Vert_{\mathrm{FKS}} \\ f \end{array}$$

In FKS there is no rule introducing new atoms bottom-up. The conclusion f is a unit and does not of course contain any atoms. So, if there is such a proof, it has to contain only units. But no rule in FKS can have a premise equivalent to t and a conclusion equivalent to f. □

The following corollary is an immediate consequence of Theorem 4.4.1 and the previous theorem:

Corollary 4.4.3. *The unit f is not provable in system SKS.*

That was consistency in a weak form. The stronger form is the following:

Theorem 4.4.4 (**Consistency**)**.** *If a formula is provable in system SKS then its negation is not provable.*

Proof. Assume that we have a proof of R and a proof of $\bar{R}$. We dualise the first proof and we get the refutation:

$$\begin{array}{c} \bar{R} \\ \Vert \\ f \end{array}$$

If we compose this derivation with the proof of $\bar{R}$ we will get a proof of f:

$$\begin{array}{c} t \\ \Vert \\ \bar{R} \\ \Vert \\ f \end{array}$$

But this is absurd because of the previous theorem. □

Conclusion

In this thesis we presented the treatment of classical logic in a deep inference formalism, the calculus of structures. We observed similarities with sequent calculus, like the admissibility of certain rules, and new properties, like symmetry of derivations and locality. Then we introduced the notions of normal proof and normal derivation and compared them with their analogues in the traditional proof-theoretic formalisms.

One of the main advantages of deep inference is the property of locality. This could be extended to other logics as well. For example, CoS was used to solve the problem of the non-locality of the promotion rule in linear logic and gave a local system for full linear logic.

For systems without involutive negation, like intuitionistic logic, we can not define implication through disjunction. Thus, we can only use two-sided sequent systems to represent them. In the calculus of structures this means introducing polarities: a context is positive if its hole corresponds to the right side of the sequent, otherwise it is negative. Tiu gives a local system for intuitionistic logic in [7].

With deep inference, we can also express in an elegant way logics that elude the traditional methods of proof theory. Many variations of modal logic, like B and K5, which are easily expressible in Frege-Hilbert systems, only find awkward presentations in sequent systems. Most of the times those systems are not analytic. Deep inference provides elegant and cut-free proof systems for modal theories. The same holds for several intermediate logics.

It is equally difficult or impossible to express proof systems for some logics involving non-commutativity and other features typical of computer science concurrency languages. After all, the original purpose of CoS was to express a logical system with a self-dual non-commutative connective resembling sequential composition in process algebras. Deep inference is really effective in this area. For example, mixed commutative/non-commutative linear logics BV and NEL are expressed in simple analytic systems. It was proved that these logics can not be expressed analytically in shallow inference.

Calculus of structures is not the only formalism employing deep inference. Another case are the nested sequents, especially targeting modal logics. The cirquent calculus, a formalism developed by Giorgi Japaridze for his computability logic, benefits from a deep inference presentation.

Bibliography

[1] A.S.Troelstra and H.Scwichtenberg. *Basic Proof Theory.* Cambridge University Press, second edition, 2000.

[2] Kai Brünnler. Two restrictions on contraction. *Logic Journal of the IGPL*, 11(5):525–529, 2003. `http://www.iam.unibe.ch/~kai/Papers/RestContr.pdf`.

[3] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs.* Logos Verlag, Berlin, 2004. `http://www.iam.unibe.ch/~kai/Papers/phd.pdf`.

[4] Kai Brünnler. Cut elimination inside a deep inference system for classical predicate logic. *Studia Logica*, 82(1):51–71, 2006. `http://www.iam.unibe.ch/~kai/Papers/q.pdf`.

[5] Kai Brünnler. Deep inference and its normal form of derivations. 3988:65–74, 2006. `http://www.iam.unibe.ch/~kai/Papers/n.pdf`.

[6] Alessio Guglielmi. Normalisation without cut elimination. `http://cs.bath.ac.uk/ag/p/AG6.pdf`, 2003.

[7] Alwen Tiu. A local system for intuitionistic logic. 4246:242–256, 2006. `http://users.cecs.anu.edu.au/~tiu/localint.pdf`.