Proofs and programs in classical logic
DRAFT
(subject to changes)

Hugo Herbelin

February 7, 2021

Contents

Part I

Proofs and programs in classical logic

Chapter 1

Introduction

1.1 The Curry-Howard isomorphism

The Curry-Howard isomorphism, also called Curry-Howard correspondence, formulae-as-types correspondence, or proofs-as-programs correspondence expresses a fundamental identity of structure between proofs and programs. It has been first hinted by Curry [Cur34, Cur42, CFC58, Sel03] who observed that simple typing for Schönfinkel's combinatory logic [Sch24, CH09] was related to Frege-Hilbert-style axiomatic systems for minimal propositional logic. It was then more subsequently developed by Howard [How69] who interpreted Gentzen's natural deduction equipped with Prawitz's normalisation procedure as an extension of Church's simply-typed λ-calculus. The main bits of the isomorphism as of the 1970's can be summarised as follows:

Frege-Hilbert systems	=	typed combinatory logic
deduction theorem	=	typed bracket abstraction
natural deduction	=	typed λ-calculus
proof normalisation	=	β-normalisation
proof	=	typed program
formula	=	type
conjunction	=	product type
disjunction	=	sum type
implication	=	function type
true formula	=	unit type
false formula	=	empty type
universal quantification	=	Π-type (or intersection type)
existential quantification	=	Σ-type (or union type)
induction	=	dependently-typed recursion
provability	=	inhabitation
"proof detour"	=	redex
proof normalisation	=	typed program normalisation
assumption	=	typed variable
inference rule	=	typed program construction rule

In 1990, a discovery by Griffin [Gri90] extended the correspondence to classical logic and control operators: classical logic opposes to intuitionistic logic in that excluded middle $A \vee \neg A$ holds by default in classical logic while, in intuitionistic logic, $A \vee \neg A$ only holds when either A or $\neg A$ is provable on its own; control operators add to pure λ-calculus the possibility to capture, restore and discard the continuation of a computation.

Griffin's discovery opened the way to a correspondence addressing more and more parts of programming and logic.

For instance, classical logic can be embedded in intuitionistic logic via so-called negative translations while control operators can be embedded into pure λ-calculus via so-called continuation-passing-style translations, leading to a new aspect of the correspondence (this was studied by Murthy [Mur90]).

Also, intuitionistic logic can be embedded in minimal logic via Friedman's A-translation [Fri78] which is connected in programming to the exception monad translation.

Similarly, Gentzen's sequent calculus can be put into correspondence with devices called abstract machines that are used to evaluate programs. This leads to an additional correspondence between the left-right symmetry of logic and a symmetry firstly between programs and their evaluation contexts, secondly between call-by-name and call-by-value evaluation.

The extended correspondence can then be summarised as follows, where one can see that classical reasoning only uses a specifically typed subset of what the programming constructs mentioned on the right-hand side are able to provide:

classical reasoning	=	control operators (with some typing)
negative translations	=	typed continuation-passing-style translations
minimal classical natural deduction	=	typed Parigot's $\lambda\mu$-calculus
classical natural deduction	=	typed $\lambda\mu tp$-calculus
sequent calculus	=	typed $\mu\tilde{\mu}$-calculus (= abstract machine)
Peirce's law	$\simeq$	Scheme's `callcc`
Ex Falso Quodlibet	$\simeq$	Felleisen's $\mathcal{A}bort$ (with some typing)
double-negation elimination	$\simeq$	Danvy-Filinski's `shift` or Felleisen's $\mathcal{C}$ (with some typing)
Friedman's A-translation	=	exception-monad translation
a form of Markov's rule	$\simeq$	# (a delimiter, with a Σ^0_1-type)
axiom of dependent choice	$\simeq$	lazy stream evaluation

An issue with classical logic is the irruption of a large variety of reduction semantics. We will also see how to interpret this flexibility by decomposing the usual look at connectives and types to the fine-grained systems of linear logic [Gir87] and further tensorial logic [MT10].

1.2 Realisability

Realisability is a relation between formulae and programs expressing that a given program "realises", i.e. "implements", a given formula. The main property is then that a program realising a formula can be extracted from any proof of that formula.

Realisability can be developed in two forms. In syntactic realisability, both the language of realisers and the logic in which realisability is expressed are object languages. For instance, the language of realisers can be Turing's machines, or recursive functions, or some variants of λ-calculus, or even, if a practical purpose is in mind, an implemented programming language such as C, OCaml, Python, Haskell, Lisp, etc. On its side, the logic in which realisability is often the same logic as the one we want to show that proofs map to programs.

In semantical realisability, the language of realisers can either be the function of the language of discourse or a specific language as above such as Turing's machine. On the other side, the property of being realisable is expressed in the language of discourse, i.e. in the metalanguage. Typical variants of semantical realisability are Tait-style reducibility [Tai67], Reynolds' parametriciy [**?**], the logical relation method [**?**] or the adequacy part of Normalisation-by-Evaluation [BS91].

The Curry-Howard correspondence can be seen as a case of syntactic realisability which looses no information contained in the proof.

On the other side, Brouwer-Heyting-Kolmogorov interpretation of proofs as an effective process is one historical instance of semantical realisability. In particular, the definition of a program is dependent on the metalanguage in which the interpretation is stated (for instance, in a classical metalanguage, control operators could be used).

In general, the semantic view at realisability can be lifted to a syntactic view by completeness theorems for realisability. Key notions of realisability includes:

- Kleene's realisability interpretation of proofs by (partial) recursive programs [Kle45]. In its syntactic presentation, it captures Heyting Arithmetic with extended Church's thesis (for instance a proof of a sentence $\forall x\, \exists y\, X(x, y)$ is interpreted as a program mapping any x to some y such that $X(x, y)$ holds).

- Gödel's functional interpretation "Dialectica" [Göd58]. In its syntactic presentation, it captures Heyting Arithmetic in finite types with the axiom of choice, Markov's principle and the principle of independence of premisses.

- Kreisel's modified realisability interpretation of proofs by simply-typed λ-terms [Kre51]. In its syntactic presentation, it captures Heyting Arithmetic in finite types with the axiom of choice and the principle of independence of premisses.

- Krivine's classical realisability which is presented semantically and whose logical strength depends on the metalanguage.

1.3 A short history of concepts

We shall have in this section an historical look at central concept of logic. We however do not claim to be as precise and exhaustive as a dedicated epistemologic work could be. In particular, part of the conclusions that we draw here are in the continuation of an approach initiated by Gentzen, and pushed further by Martin-Löf, Girard, Coquand, ...

1.3.1 Formulae

The idea about whether a formula is a sequence of symbols or a tree varied. Formulae are graphical trees in Frege, but Church (1944) made popular the idea that formulae are strings of symbols out of which only *well-formed formulae* are the interesting ones, taking with him several American authors of textbooks.

In any cases, writing a tree linearly needs some convention: parentheses and rules of precedence, or e.g. either the Polish or reverse Polish notation, etc. With the coming of computer science, the duality between the concept and its representation has somehow been resolved with the distinction between abstract and concrete syntax tree, the former corresponding to the concept, namely a tree, concentrating on the nodes and leaves, but abstracting on the use of parentheses and precedences which are relevant to the concrete syntax, even though even an abstract syntax needs a concrete formal support to be represented - one cannot escape syntax! -.

- Boole introduced the formal notion of propositional "formula" and of its arithmetical interpretation over $\{0,1\}$

$$A \quad ::= \quad X \;\mid\; A \wedge A \mid A \vee A$$

- Peirce introduced the formal notion of quantification and of the formulae of what is now known as predicate calculus, with a semantic approach interpreting formulae as relations

$$A \quad ::= \quad R(x,y) \;\mid\; A \wedge A \mid A \vee A \mid \forall x\, A \mid \exists x\, A$$

- Frege introduced independently of Peirce the notion of formula and predicate calculus but together with the formal notion of (axiomatic) proof.

1.3.2 Proofs

The same question holds for proofs. For Frege, who introduced proofs as a formal mathematical object, proofs were sequences of proof steps. On the other side Gentzen introduced the idea that a proof is a structured object taking the form of a tree, represented graphically (see however [Pel99] for an history of natural deduction, hence implicitly of the genesis of the idea that a proof is a structured tree).

1.3.3 Judgements

In Frege-Hilbert proof systems (see Chapter 4), a formula is derived from hypotheses and axioms using two inference rules, modus ponens and generalisation. Implication and universal quantification come first and the property of other connectives are axiomatised. E.g., the three axioms $A \Rightarrow B \Rightarrow A \wedge B$, $A \wedge B \Rightarrow A$ and $A \wedge B \Rightarrow B$ express the properties of conjunction.

Starting from Gentzen comes the concept of hypothetical judgement which itself lays on the technical concept of sequent, written $\Gamma \vdash A$, i.e. of assertion of a formula under some context of hypotheses Γ. Using sequents, it becomes possible to define each connective intrinsically from its properties using inference rules. For instance, the conjunction can be characterised by rules rather than axioms, as follows:

$$\wedge_E^1 \ \frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_1} \qquad \wedge_E^2 \ \frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_2} \qquad \wedge_I \ \frac{\Gamma \vdash A_1 \qquad \Gamma \vdash A_2}{\Gamma \vdash A_1 \Rightarrow A_2}$$

While Gentzen distinguishes between the concepts of implication and judgement, Frege-Hilbert approach uses implication as an internalisation of the notion of judgement.

In Gentzen, implication can be seen as a reification of the notion of judgement as a connective, what the following inference rules for implication express:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow_E$$

On the other side, in Frege-Hilbert's systems, only a limited usage of the concept of hypothetical judgement is used, for which the set Γ is fixed throughout the derivation. This is by the way witnessed by the "deduction theorem" in Frege-Hilbert's systems: If B is provable under hypothesis A, then $A \Rightarrow B$ is provable. This is proved by induction on the derivation of B, i.e. it is an admissible rule, while in Gentzen this is defining inference rule for implication.

In distinguishing the notion of sequent from the notion of implication, Gentzen allowed a notion of "purity of methods" highlighting the intrinsic properties of logical connectives.

1.4 Historical notes

A history of intuitionism has been written by Troelstra [Tro91].

Part of this history also includes the observation by De Bruijn that proofs of his AUTOMATH's type theory could be represented by λ-terms [dB68].

The Curry-Howard correspondence spread to category theory with Lambek's connection between propositional logic, combinatory logic and Cartesian closed categories [Lam72].

1.5 Conventions

Our approach is in a large part syntactic, so we shall define in a large part algebraic structures in the form of data-types (using a programming terminology), i.e. initial algebra (using a category theory terminology), i.e. inductive types (using a type-theoretic terminology). We shall use Backus-Naur style to define the concrete grammars for such algebraic structures, as e.g. in:

$$A, B \quad ::= \quad X \mid A \Rightarrow B$$

We shall indicate that we extend a BNF grammar by using the $\ldots$ as e.g. in:

$$A, B \quad ::= \quad \ldots \mid A \wedge B$$

We shall use inference rules to define algebraic structures with constraints, i.e. Generalised Abstract Data Types (using a terminology from programming language) or inductive families (using a type-theoretic terminology), as e.g. the definition of $\vdash$ in:

$$\Rightarrow_I \quad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

Chapter 2

Classical logic vs intuitionistic logic

Until questioned by Brouwer around the beginning of 20^{th} century, the classical view on logic was that any sentence, i.e. closed formula, was either true or false. At the level of provability, this view justifies the axiom of excluded middle, i.e. the scheme $A \vee \neg A$ for any (closed or not closed) formula A. This is a scheme which does not tell which of A or $\neg A$ is true, but a scheme which always allows to assume that an arbitrary proposition is true or not during a proof.

Brouwer defended the idea that for a sentence to be considered "true" an effective justification could be given. E.g. an existential statement $\exists x\, A(x)$ should be "true" only when an effective witness can be given and a disjunctive statement should be "true" only when there is an evidence that one of the two disjuncts is effectively true. For instance, from this point of view, a statement of the form $\forall x\, (A(x) \Rightarrow \exists y\, B(x,y))$ should be evidenced by a function returning a y satisfying $B(x,y)$ for all x satisfying $A(x)$. Brouwer's view rejects $A \vee \neg A$ as a general scheme, arguing that we cannot assert a priori having a proof of either A or $\neg A$ without having an effective proof of one side of the disjunction. This position was later on formally justified by Church and Turing's undecidability of provability in first-order logic and arithmetic: there is no effective process telling whether A or $\neg A$ holds for an arbitrary A.

This alternative view on formal logic was called intuitionistic logic. Contrastingly, the logic accepting excluded middle as a general scheme inherited the name of "classical logic".

For Brouwer, intuitionistic logic restricts the class of formulae that classical logic can prove, since not all instances of excluded-middle are provable. Kolmogorov showed however that statements provable in classical logic are provable in intuitionistic logic if one accepts to reason modulo double-negation. In particular, classical logic is embeddable into intuitionistic logic via so-called negative translations which prefix positive connectives by a double negation. This shows that intuitionistic logic actually gives to the positive connectives $\vee$ and $\exists$, as well as to atoms, a more informative meaning:

- Proving $A \vee B$ in intuitionistic logic means giving an effective proof of A or an effective proof of B. Similarly, proving $\exists x\, A(x)$ means giving an effective witness t such that $A(t)$ holds
- Proving $A \vee B$ in classical logic only means giving an intuitionistic proof of $\neg\neg(A \vee B)$, which is weaker since, intuitionistically $A \vee B$ implies $\neg\neg(A \vee B)$ but not the converse (and similarly for $\exists x\, A(x)$).

We will look at this in more details in the next sections.

2.1 Intuitionistic natural deduction

Let us recall the definition of intuitionistic natural deduction in "sequent" style, i.e. as a system deriving *sequents* (not to be confused with *sequent calculus* which is considered in Chapter 9). Terms are built from the following grammar:

$$t, u \quad ::= \quad x \mid f(t_1, ..., t_{\mathsf{ar}_f})$$

where x ranges over an countable set of term variables and f ranges over an alphabet of function symbols (each symbol comes with an arity ar_f and takes the corresponding number of terms as arguments). Formulae are built from the following grammar:

$$A, B, C \quad ::= \quad X(t_1, ..., t_{\mathsf{ar}_X}) \mid \top \mid \bot \mid A \vee A \mid A \wedge A \mid A \Rightarrow A \mid \exists x\, A \mid \forall x\, A$$

where X ranges over an alphabet of predicate symbols (each symbol comes again with an arity ar_X and takes the corresponding number of terms as arguments). In $\forall x\, A$ and in $\exists x\, A$, x binds the instances of x occurring in the position of a term in A (e.g. in $\exists x\, (X(x) \Rightarrow Y(x))$, the x in $\exists x$ binds the two occurrences of x in $X(x)$ and $Y(x)$). If a variable x occurs in position of a term in A, without being in the scope of a $\forall x$ or of an $\exists x$ inside A, one says that the variable *occurs free* in A. Binders are considered up to α-conversion, i.e. up to the exact names used for binding

as soon as the structure of bindings is preserved (for instance, we do not make a difference between $\forall a, X(a) \Rightarrow Y(a)$ and $\forall b, X(b) \Rightarrow Y(b)$). By $A[t/x]$ we denote the replacement of every free occurrence of x in A by t (possibly renaming bound variables to prevent capture of variables in t, i.e. reasoning modulo α-conversion).

In writing formulae, as well as in writing most all syntactic objects defined hereafter, we will use parentheses to group subobjects. For instance, we write $(A \wedge B) \vee C$ to mean the formula whose structure has $\vee$ has main connective and $\wedge$ as main connective of the left-hand subformula of the initial formula. However, we use precedences to reduce the need for parentheses. The order of precedence we use is the following: $\forall$ and $\exists$ bind more than $\wedge$ which binds more than $\vee$ which binds more than $\Rightarrow$. To reduce parentheses, we shall also use associativity rules: $\vee$, $\wedge$ and $\Rightarrow$ shall associate by default to the right, meaning that $A \vee B \vee C$, $A \wedge B \wedge C$ and $A \Rightarrow B \Rightarrow C$ shall mean $A \vee B \vee C$, $A \wedge B \wedge C$ and $A \Rightarrow B \Rightarrow C$ shall mean $A \vee (B \vee C)$, $A \wedge (B \wedge C)$ and $A \Rightarrow (B \Rightarrow C)$.

The connectives $\vee$ and $\exists$ are called *positive*. The connectives $\Rightarrow$ and $\forall$ are *negative*. Conjunction, as well as the true and false formulae, can be considered either as negative or positive, depending on the purpose. Atoms are considered to be positive unless stated otherwise.

The following definition of negation in terms of implication and absurdity is standard (in writing, $\neg$ will have the same precedence as $\forall$ and $\exists$):

$$\neg A \quad \triangleq \quad A \Rightarrow \bot$$

Contexts are (ordered) lists of formulae, as described by the following grammar:

$$\Gamma, \Delta \quad ::= \quad \emptyset \mid \Gamma, A$$

Remark 1 *Using lists of formulae rather than sets or multisets is a necessary condition to have a non trivial proofs-as-programs correspondence. By using sets or multisets, there would be no way to distinguish the two possible "programs", i.e. normal proofs, of type $A \Rightarrow A \Rightarrow A$, as the two assumptions of A would be indistinguishable. On the contrary, by using lists, the formulae are distinguished by their position*[1]*.*

Juxtaposition of contexts and formulae, separated with a comma, as in Γ, A, Δ denotes the list concatenation of a sequence of contexts and of formulae, with lonely formulae being interpreted as singleton contexts. Note that the notation Γ, A can be syntactically interpreted in two different ways (i.e. as the context Γ, A or as the concatenation of Γ and of the singleton context A) but both interpretations coincide. We say that a variable x occurs free in Γ if it occurs free in at least one of the formulae of Γ.

We may apply transformations to contexts. For instance $\neg\Gamma$ will mean the context obtained by replacing all formulae A in Γ by $\neg A$.

The rules concern sequents, where an intuitionistic sequent, written $\Gamma \vdash_I A$, expresses that the conclusion A holds under the assumption of the hypotheses in Γ (the "entails" symbol $\vdash$ originates from Frege's graphical proofs [Fre67] and its generalisation by a context of hypotheses appears in Gentzen [Gen35]).

To each connective is associated introduction rules (written with a subscript $_I$) and elimination rules (written with a subscript $_E$). Those elimination rules which have several premisses have a main premisse which refers to the corresponding connective. There is an additional purely structural rule[2]. Altogether, the rules of intuitionistic natural deduction for predicate logic are given in Figure 2.1.

We will omit the writing of $\emptyset$ in judgements of the form $\emptyset \vdash_I A$, and hence write $\vdash_I A$ instead.

When referring to an inference rule, the premisses of the rule are the sequents appearing above the line and the conclusion of the rule is the sequent appearing below the line.

A derivation of $\Gamma \vdash_I A$ is a tree of inference rules, with the premisses of a rule matching the conclusion of the children rules and with the rule Ax at the leaves of the tree. A tree is an object over which one can reason by structural induction, with the rule Ax as base case and the other rules as inductive cases.

A rule is said *derivable* if there is a tree of inference rules whose sequents at the leaves are the premisses of the derived rule and the sequent conclusion of the tree is the sequent conclusion of the rule.

A rule is said *admissible* if for every derivation of the sequent premisses of the rule, one can find a derivation of the sequent conclusion of the rule.

Minimal natural deduction is intuitionistic natural deduction without the rule $\bot_E$ (i.e. Ex Falso Quodlibet). We write $\Gamma \vdash_M A$ to denote a derivation which does not use $\bot_E$.

We now state an important technical lemma about proofs which is proved by induction on the structure of the proof.

Lemma 1 (Admissibility of weakening) *If $\Gamma_1, \Gamma_2 \vdash A$ then $\Gamma_1, B, \Gamma_2 \vdash A$.*

[1] An alternative is to use sets of *named* formulae, as commonly done when typing λ-calculus. But we prefer to use lists, so as to have a formulation looking more like the standard formulation of logic.

[2] Some authors, such as [GLT89], restrict the terminology *structural rule* to the rules of weakening, contraction, and, when relevant, exchange, referring instead to the axiom rule as an identity rule. In Gentzen's sequent calculus, the axiom rule is also treated seperatedly from other structural rules, being called the *initial rule*. We shall instead call *structural* all rules which are not specifying a particular connective.

structural rule

$$\frac{}{\Gamma_1, A, \Gamma_2 \vdash_I A}\ Ax$$

rules defining connectives and quantifiers

$$\frac{}{\Gamma \vdash_I \top}\ \top_I \qquad \frac{\Gamma \vdash_I \bot}{\Gamma \vdash_I A}\ \bot_E$$

$$\frac{\Gamma \vdash_I A \quad \Gamma \vdash_I B}{\Gamma \vdash_I A \wedge B}\ \wedge_I \qquad \frac{\Gamma \vdash_I A \wedge B}{\Gamma \vdash_I A}\ \wedge_E^1 \qquad \frac{\Gamma \vdash_I A \wedge B}{\Gamma \vdash_I B}\ \wedge_E^2$$

$$\frac{\Gamma \vdash_I A}{\Gamma \vdash_I A \vee B}\ \vee_I^2 \qquad \frac{\Gamma \vdash_I B}{\Gamma \vdash_I A \vee B}\ \vee_I^2 \qquad \frac{\Gamma \vdash_I A \vee B \quad \Gamma, A \vdash_I C \quad \Gamma, B \vdash_I C}{\Gamma \vdash_I C}\ \vee_E$$

$$\frac{\Gamma, A \vdash_I B}{\Gamma \vdash_I A \Rightarrow B}\ \Rightarrow_I \qquad \frac{\Gamma \vdash_I A \Rightarrow B \quad \Gamma \vdash_I A}{\Gamma \vdash_I B}\ \Rightarrow_E$$

$$\frac{\Gamma \vdash_I A \quad x \text{ not occurring free in } \Gamma}{\Gamma \vdash_I \forall x\, A}\ \forall_I \qquad \frac{\Gamma \vdash_I \forall x\, A}{\Gamma \vdash_I A[t/x]}\ \forall_E$$

$$\frac{\Gamma \vdash_I A[t/x]}{\Gamma \vdash_I \exists x\, A}\ \exists_I \qquad \frac{\Gamma \vdash_I \exists x\, A \quad \Gamma, A \vdash_I C \quad x \text{ not occurring free in } \Gamma \text{ and } C}{\Gamma \vdash_I C}\ \exists_E$$

Figure 2.1: Intuitionistic natural deduction

PROOF: By induction on the derivation of $\Gamma_1, \Gamma_2 \vdash A$, extending Γ_2 each time a $\Rightarrow_I$ rule, or $\vee_E$ or $\exists_E$ is used. In particular, the weakening is eventually absorbed at the level of axioms. ■

Note that we also have the following lemma (in general technically less useful though), which is also proved by induction on the structure of the proof.

Lemma 2 (Admissibility of contraction) *If* $\Gamma_1, A, \Gamma_2, A, \Gamma_3 \vdash B$ *then* $\Gamma_1, A, \Gamma_2, \Gamma_3 \vdash B$ *and* $\Gamma_1, \Gamma_2, A, \Gamma_3 \vdash B$

PROOF: By induction on the derivation of $\Gamma_1, A, \Gamma_2, A, \Gamma_3 \vdash B$, extending Γ_3 each time a $\Rightarrow_I$ rule, or $\vee_E$ or $\exists_E$ is used, and, in the axiom rule, by changing the references to a removed designated A into a reference to the remaining designated A. ■

Remark 2 *There are alternative presentations of natural deduction where contraction and weakening are explicit inference rules, called* structural *rules. The presentations are equivalent in terms of provability, but the one we give is better-suited for a proofs-as-programs interpretation in that the context* Γ *in the premisses of each rule is canonically derivable from the information contained in the conclusion of the rule.*

Remark 3 *Two presentations are possible for conjunction. The presentation above defined a negative conjunction. We can define a positive conjunction with the following rules:*

$$\frac{\Gamma \vdash_I A \quad \Gamma \vdash_I B}{\Gamma \vdash_I A \wedge B}\ \wedge'_I \qquad \frac{\Gamma \vdash_I A \wedge B \quad \Gamma, A, B \vdash_I C}{\Gamma \vdash_I C}\ \wedge'_E$$

The difference of presentation echoes to the difference between the two conjunctions & *(a negative conjunction) and* $\otimes$ *(a positive conjunction) in linear logic.*

This distinction applies to the true formula as well. The presentation above defined a negative true formula, which can be seen as the neutral element of the negative conjunction, i.e. as an empty negative conjunction. Even if of little interest, we can define a positive true formula with the following rules:

$$\frac{}{\Gamma \vdash_I \top}\ \top'_I \qquad \frac{\Gamma \vdash_I \top \qquad \Gamma \vdash_I C}{\Gamma \vdash_I C}\ \top'_E$$

This defines the true formula as the neutral element of the positive conjunction, i.e. as the empty positive conjunction.

Remark 4 *There is an alternative presentation of intuitionistic logic which uses two kinds of sequent:* $\Gamma \vdash_I A$ *as in the above presentation, but also sequents of the form* $\Gamma \vdash_I$ *with no particular conclusion (such a sequent can be thought of as* Γ *is contradictory). Then the rule for* $\bot$ *is replaced by the following two rules:*

$$\frac{\Gamma \vdash_I \bot}{\Gamma \vdash_I}\ \bot'_E \qquad \frac{\Gamma \vdash_I}{\Gamma \vdash_I A}\ weak_R$$

The difference of presentation is loosely related to the difference between the two "false" formulae $\bot$ *(a negative formula) and* 0 *(a positive formula, which can be seen as the neutral element of disjunction, i.e. as an empty disjunction) in linear logic, where the first presentation defines the positive false formula (i.e.* 0 *in the sense of linear logic) and the alternative presentation loosely defines the negative false formula(i.e.* $\bot$ *in the sense of linear logic). The reason why we only say "loosely related" will become clearer after having presented the classical case.*

2.2 Classical natural deduction

Gentzen defined natural deduction for classical logic by extending intuitionistic natural deduction with an axiom

$$\frac{}{\Gamma \vdash A \vee \neg A}$$

We define classical natural deduction instead by allowing multiple formulae on the right-hand side of a sequent, as introduced by Parigot [Par91], and as initially developed by Gentzen in the context of sequent calculus LK. This provide with a purely structural hence more essential view at classical logic independent of the existence of this or that connective. *Classical natural deduction* manipulates two kinds of sequents:

$\Gamma \vdash_C \Delta;$	passive sequents (also called commands)
$\Gamma \vdash_C \Delta; A$	active sequents

structural rules

$$\frac{}{\Gamma_1, A, \Gamma_2 \vdash_C \Delta; A}\ Ax$$

$$\frac{\Gamma \vdash_C \Delta_1, A, \Delta_2; A}{\Gamma \vdash_C \Delta_1, A, \Delta_2;}\ \textit{Unfocus} \qquad \frac{\Gamma \vdash_C \Delta, A;}{\Gamma \vdash_C \Delta; A}\ \textit{Focus}$$

rules defining connectives and quantifiers

$$\frac{}{\Gamma \vdash_C \Delta; \top}\ \top_I \qquad \frac{\Gamma \vdash_C \Delta; \bot}{\Gamma \vdash_C \Delta; A}\ \bot_E$$

$$\frac{\Gamma \vdash_C \Delta; A \quad \Gamma \vdash_C \Delta; B}{\Gamma \vdash_C \Delta; A \wedge B}\ \wedge_I \qquad \frac{\Gamma \vdash_C \Delta; A \wedge B}{\Gamma \vdash_C \Delta; A}\ \wedge_E^1 \qquad \frac{\Gamma \vdash_C \Delta; A \wedge B}{\Gamma \vdash_C \Delta; B}\ \wedge_E^2$$

$$\frac{\Gamma \vdash_C \Delta; A}{\Gamma \vdash_C \Delta; A \vee B}\ \vee_I^1 \qquad \frac{\Gamma \vdash_C \Delta; B}{\Gamma \vdash_C \Delta; A \vee B}\ \vee_I^2 \qquad \frac{\Gamma \vdash_C \Delta; A \vee B \quad \Gamma, A \vdash_C \Delta; C \quad \Gamma, B \vdash_C \Delta; C}{\Gamma \vdash_C \Delta; C}\ \vee_E$$

$$\frac{\Gamma, A \vdash_C \Delta; B}{\Gamma \vdash_C \Delta; A \Rightarrow B}\ \Rightarrow_I \qquad \frac{\Gamma \vdash_C \Delta; A \Rightarrow B \quad \Gamma \vdash_C \Delta; A}{\Gamma \vdash_C \Delta; B}\ \Rightarrow_E$$

$$\frac{\Gamma \vdash_C \Delta; A[y/x] \quad y \text{ not occurring free in } \Gamma}{\Gamma \vdash_C \Delta; \forall x\, A}\ \forall_I \qquad \frac{\Gamma \vdash_C \Delta; \forall x\, A}{\Gamma \vdash_C \Delta; A[t/x]}\ \forall_E$$

$$\frac{\Gamma \vdash_C \Delta; A[t/x]}{\Gamma \vdash_C \Delta; \exists x\, A}\ \exists_I \qquad \frac{\Gamma \vdash_C \Delta; \exists x\, A \quad \Gamma, A[y/x] \vdash_C \Delta; C \quad y \text{ not occurring free in } \Gamma}{\Gamma \vdash_C \Delta; C}\ \exists_E$$

Figure 2.2: Classical natural deduction (Parigot's style)

The rules are those of intuitionistic natural deduction with an extra context of conclusions Δ on the right, plus two rules to deal with the formulae in Δ, as shown in Figure 2.2.

We shall simply write $\Gamma \vdash_C A$ for $\Gamma \vdash_C ; A$ and sometimes $\Gamma \vdash_C \Delta$ for $\Gamma \vdash_C \Delta;$.

As in the intuitionistic case, weakening and contraction are admissible in the context of hypotheses. In addition, they are admissible also in the context of conclusions on the right-hand side.

Lemma 3 (Admissibility of weakening on the right)
If $\Gamma \vdash \Delta_1, \Delta_2; A$ *then* $\Gamma \vdash \Delta_1, B, \Delta_2; A$. *If* $\Gamma \vdash \Delta_1, \Delta_2;$ *then* $\Gamma \vdash \Delta_1, B, \Delta_2;$.

PROOF: By induction on the derivation of $\Gamma \vdash \Delta_1, \Delta_2; A$ (resp. $\Gamma \vdash \Delta_1, \Delta_2;$), extending Δ_2 each time a *Save* rule is used. In particular, the weakening is eventually absorbed at the level of axioms. ■

Lemma 4 (Admissibility of contraction on the right)
If $\Gamma \vdash \Delta_1, B, \Delta_2, B, \Delta_3; A$ *then* $\Gamma \vdash \Delta_1, B, \Delta_2, \Delta_3; A$ *and* $\Gamma \vdash \Delta_1, \Delta_2, B, \Delta_3; A$. *If* $\Gamma \vdash \Delta_1, B, \Delta_2, B, \Delta_3;$ *then* $\Gamma \vdash \Delta_1, B, \Delta_2, \Delta_3;$ *and* $\Gamma \vdash \Delta_1, \Delta_2, B, \Delta_3;$.

PROOF: By induction on the derivation of $\Gamma \vdash \Delta_1, B, \Delta_2, B, \Delta_3; A$ (resp. $\Gamma \vdash \Delta_1, B, \Delta_2, B, \Delta_3;$), extending Δ_3 each time a *Save* rule is used, and, in rule *Unfocus*, by changing the references to a removed designated B into a reference to the remaining designated B. ■

Remark 5 *Unfocus and Focus are sometimes called Pass and Act respectively, to be interpreted in a top-down reading of the derivation. Alternative names also include Restore and Save as a reference to the computational action of these rules on the surrounding evaluation context.*

Remark 6 *The extra inference rules for positive conjunction and positive true formula adapt canonically. The alternative inference rules for the negative false formula can however be reformulated in a smoother way by exploiting the possibility for several conclusions:*

$$\frac{\Gamma \vdash_C \Delta; \bot}{\Gamma \vdash_C \Delta;} \bot'_E \qquad \frac{\Gamma \vdash_C \Delta;}{\Gamma \vdash_C \Delta; \bot} \bot'_I$$

Exercice 1 *Give a proof of Excluded-Middle ($A \vee \neg A$ for arbitrary A), Double-Negation Elimination ($\neg\neg A \Rightarrow A$) and Peirce's law ($((A \Rightarrow B) \Rightarrow A) \Rightarrow A$) in this system.*

Chapter 3

The proofs-as-programs interpretation of natural deduction

Historically, the Curry-Howard isomorphism is the observation of a correspondence between typed combinatory logic and Frege-Hilbert axiomatic systems for propositional logic by Curry from 1930, followed by the observation in 1969 that natural deduction can be seen as a simply-typed λ-calculus by Howard[1], with the normalisation of proofs matching the normalisation of λ-terms[2]. Curry's correspondence between Frege-Hilbert systems and combinatory logic will be addressed in Chapter 4. We shall here address the correspondence between λ-calculus and natural deduction.

3.1 The intuitionistic case

To highlight the computational content of proofs, we annotate every derivation of a sequent $\Gamma \vdash_I A$ by a program p reflecting the structure of the proof. We write $\Gamma \vdash_I p : A$ for an annotated sequent.

To make human reading easier, we also annotate the formulae in Γ by names ranging over a set of variables a, b, c, ... This is not strictly necessary, though, since one could also refer to formulae of Γ by position.

In the intuitionistic case, we annotate sequents by *terms*, i.e. *programs*, defined by the following grammar, which follows the structure of proofs:

$$\begin{array}{rcl} p, q, r & ::= & a \mid () \mid \mathsf{efq}\, p \mid (p, q) \mid \pi_1(p) \mid \pi_2(p) \mid \iota_1(p) \mid \iota_2(p) \mid \mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r] \\ & \mid & \lambda a.p \mid p\, q \mid \lambda x.p \mid p\, t \mid (t, p) \mid \mathsf{dest}\ p\ \mathsf{as}\ (x, a)\ \mathsf{in}\ q \end{array}$$

In $\lambda a.p$ and $\mathsf{dest}\ p\ \mathsf{as}\ (x, a)\ \mathsf{in}\ q$, a is a binder of proof. It binds the occurrences of a occurring in p and q respectively. In $\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r]$, a and b are also binders of proofs. In $\lambda x.p$ and $\mathsf{dest}\ p\ \mathsf{as}\ (x, a)\ \mathsf{in}\ q$, x is a binder of term. As for formulae, we reason up to the exact choice of the name used for binding. We write $FV(p)$ for the free variables of p respectively. This is straightforwardly defined inductively by

$$\begin{array}{ll} FV(a) & \triangleq \{a\} \\ FV(()) & \triangleq \emptyset \\ FV(\mathsf{efq}\, p) & \triangleq FV(p) \\ FV((p, q)) & \triangleq FV(p) \cup FV(q) \\ FV(\pi_1(p)) & \triangleq FV(p) \\ FV(\pi_2(p)) & \triangleq FV(p) \\ FV(\iota_1(p)) & \triangleq FV(p) \\ FV(\iota_2(p)) & \triangleq FV(p) \\ FV(\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r]) & \triangleq FV(p) \cup (FV(q) \backslash \{a\}) \cup (FV(r) \backslash \{b\}) \\ FV(\lambda a.p) & \triangleq FV(p) \backslash \{a\} \\ FV(p\, q) & \triangleq FV(p) \cup FV(q) \\ FV(\lambda x.p) & \triangleq FV(p) \\ FV(p\, t) & \triangleq FV(p) \\ FV((t, p)) & \triangleq FV(p) \\ FV(\mathsf{dest}\ p\ \mathsf{as}\ (x, a)\ \mathsf{in}\ q) & \triangleq FV(p) \cup (FV(q) \backslash \{a\}) \end{array}$$

[1] Howard did not consider natural deduction strictly speaking in that only implication and universal quantification are defined by the inference rules of natural deduction; the other connectives are instead defined axiomatically, as in Frege-Hilbert's systems.

[2] See Troelstra [Tro99] for a rather comprehensive view at the genesis of the proofs-as-programs correspondence, or, more recently [?]. In particular, De Bruijn can be mentioned as he independently used λ-calculus to denote proofs of the AUTOMATH system [dB68].

structural rule

$$\frac{}{\Gamma_1, a : A, \Gamma_2 \vdash_I a : A}\ Ax$$

rules defining connectives and quantifiers

$$\frac{}{\Gamma \vdash_I () : \top}\ \top_I \qquad \frac{\Gamma \vdash_I p : \bot}{\Gamma \vdash_I \mathsf{efq}\, p : A}\ \bot_E$$

$$\frac{\Gamma \vdash_I p : A \quad \Gamma \vdash_I q : B}{\Gamma \vdash_I (p, q) : A \wedge B}\ \wedge_I \qquad \frac{\Gamma \vdash_I p : A \wedge B}{\Gamma \vdash_I \pi_1(p) : A}\ \wedge_E^1 \qquad \frac{\Gamma \vdash_I p : A \wedge B}{\Gamma \vdash_I \pi_2(p) : B}\ \wedge_E^2$$

$$\frac{\Gamma \vdash_I p : A}{\Gamma \vdash_I \iota_1(p) : A \vee B}\ \vee_I^2 \qquad \frac{\Gamma \vdash_I p : B}{\Gamma \vdash_I \iota_2(p) : A \vee B}\ \vee_I^2 \qquad \frac{\Gamma \vdash_I p : A \vee B \quad \Gamma, a : A \vdash_I q : C \quad \Gamma, b : B \vdash_I r : C}{\Gamma \vdash_I \mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r] : C}\ \vee_E$$

$$\frac{\Gamma, a : A \vdash_I p : B}{\Gamma \vdash_I \lambda a.p : A \Rightarrow B}\ \Rightarrow_I \qquad \frac{\Gamma \vdash_I p : A \Rightarrow B \quad \Gamma \vdash_I q : A}{\Gamma \vdash_I p\, q : B}\ \Rightarrow_E$$

$$\frac{\Gamma \vdash_I p : A \quad x \text{ not occurring free in } \Gamma}{\Gamma \vdash_I \lambda x.p : \forall x\, A}\ \forall_I \qquad \frac{\Gamma \vdash_I p : \forall x\, A}{\Gamma \vdash_I p\, t : A[t/x]}\ \forall_E$$

$$\frac{\Gamma \vdash_I p : A[t/x]}{\Gamma \vdash_I (t, p) : \exists x\, A}\ \exists_I \qquad \frac{\Gamma \vdash_I p : \exists x\, A \quad \Gamma, a : A \vdash_I q : C \quad x \text{ not occurring free in } \Gamma \text{ and } C}{\Gamma \vdash_I \mathsf{dest}\ p\ \mathsf{as}\ (x, a)\ \mathsf{in}\ q : C}\ \exists_E$$

Figure 3.1: Intuitionistic natural deduction (annotated)

We similarly define the free term variables of p by:

$$\begin{array}{lcl} FV_t(\lambda x.p) & \triangleq & FV_t(p) \backslash \{x\} \\ FV_t(p\, t) & \triangleq & FV_t(t) \cup FV_t(p) \\ FV_t((t, p)) & \triangleq & FV_t(t) \cup FV_t(p) \\ FV_t(\mathsf{dest}\ p\ \mathsf{as}\ (x, a)\ \mathsf{in}\ q) & \triangleq & FV_t(p) \cup (FV_t(q) \backslash \{x\}) \end{array}$$

with all other clauses as in $FV(p)$.

We use names to annotate hypotheses, hence contexts are redefined using the following grammar:

$$\Gamma \quad ::= \quad \emptyset \mid \Gamma, a : A$$

and sequents now have the form $\Gamma \vdash_I p : A$ where p reflects the structure of the proof. We expect that in Γ, all names are distinct. The annotated system is given on Figure 3.1.

Note that the invariant that all names are distinct in Γ expects that in the rules adding an hypothesis to Γ, a name for the binder is chosen such that it is different from the names already used.

We can now state the main property of the annotations. Since they reflect the structure of the proof, we have the following which we state as a formal instance of the "Curry-Howard isomorphism":

Property 1 (Curry-Howard isomorphism) *Let Γ be an annotated context and $|\Gamma|$ its non annotated form. If π is a derivation of $|\Gamma| \vdash_I A$, there is a unique program $extract(\pi)$ up to the choice of names such that $\Gamma \vdash_I extract(\pi) : A$ holds. Conversely, to any program p such that $\Gamma \vdash_I p : A$ holds, there is a unique derivation $proof_{\Gamma,A}(p)$ of $|\Gamma| \vdash_I A$ such that $extract(proof_{\Gamma,A}(p)) = p$ and, for π a derivation of $|\Gamma| \vdash_I A$, $proof_{\Gamma,A}(extract(\pi)) = \pi$. Moreover, both maps respect the structure of the inference rules.*

In particular, this means that any reasoning about proofs can be made over the underlying typed program. For instance, normalisation of proofs can be expressed using a rewriting system over terms, which preserves typing, as will be done in the next section.

Remark 7 *Annotating contexts with variables is not essential. Numerical indices referring to the position of the hypothesis in the list could be used instead. This is known as* De Bruijn indices *(when numbering from the right) [dB78], or* De Bruijn levels *(when numbering from the left) and it is similar to the kind of indices used in compilers. However, this makes things less easy to read for humans which is why we prefer using names here.*

We rephrase the weakening and contraction lemmas on annotated proofs. The proofs are similar as in the non-annotated case.

Lemma 5 (Admissibility of weakening)
If $\Gamma_1, \Gamma_2 \vdash p : A$ and b is not already used in Γ_1, Γ_2, then $\Gamma_1, b : B, \Gamma_2 \vdash p : A$.
If $\Gamma \vdash \Delta_1, \Delta_2; p : A$ and β is not already used in Δ_1, Δ_2, then $\Gamma \vdash \Delta_1, \beta : B, \Delta_2; p : A$.

Lemma 6 (Admissibility of contraction)
If $\Gamma_1, a_1 : A, \Gamma_2, a_2 : A, \Gamma_3 \vdash p : B$ then $\Gamma_1, a_1 : A, \Gamma_2, \Gamma_3 \vdash p[a_1/a_2] : B$ and $\Gamma_1, \Gamma_2, a_2 : A, \Gamma_3 \vdash p[a_2/a_1] : B$.

We also have a strengthening lemma.

Lemma 7 (Admissibility of strengthening)
If $\Gamma_1, a : A, \Gamma_2 \vdash p : B$ and $a \notin FV(p)$ then $\Gamma_1, \Gamma_2 \vdash p : B$.

PROOF: By induction on the derivation of $\Gamma_1, a : A, \Gamma_2 \vdash \Delta; p : B$. That a does not occur in p ensures that the case of an axiom rule annotated by a is impossible. ■

3.2 Normalisation of intuitionistic natural deduction (call-by-name case)

To define normalisation in natural deduction, it is convenient to introduce the notion of elementary evaluation contexts, below, using letter F to ranged over them:

$$F \quad ::= \quad [\,]\,p \;\mid\; \pi_1([\,]) \;\mid\; \pi_2([\,]) \;\mid\; \mathsf{case}\ [\,]\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r] \;\mid\; \mathsf{dest}\ [\,]\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q \;\mid\; \mathsf{efq}([\,]) \;\mid\; [\,]\,t$$

In the definition of F, the notation $[\,]$ denotes a hole to be filled by a program. We use the meta-notation $F[p]$ to mean the term obtained by plugging p into the hole of F.

Reduction rules for intuitionistic natural deduction proofs can then be defined by the following *rewriting system* where we distinguish between *computational rules*, also called *logical reduction rules*, which contract the interaction between an introduction and an elimination of the same connective by removing the need for the connective, and *commutative cuts*, also called *commutative conversions* or *permutative conversions*, which move elimination rules across those elimination rules for positive connectives which may block the applicability of computational rules.

computational rules

$$\begin{array}{lll}
(\beta_{\Rightarrow}) & (\lambda a.p)q & \xrightarrow{h} \; p[q/a] \\
(\beta^1_{\wedge}) & \pi_1(p,q) & \xrightarrow{h} \; p \\
(\beta^2_{\wedge}) & \pi_2(p,q) & \xrightarrow{h} \; q \\
(\beta^1_{\vee}) & \mathsf{case}\ \iota_1(p)\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r] & \xrightarrow{h} \; q[p/a] \\
(\beta^2_{\vee}) & \mathsf{case}\ \iota_2(p)\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r] & \xrightarrow{h} \; r[p/b] \\
(\beta_{\exists}) & \mathsf{dest}\ (t,p)\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q & \xrightarrow{h} \; q[t/x][p/a] \\
(\beta_{\forall}) & (\lambda x.p)t & \xrightarrow{h} \; p[t/x]
\end{array}$$

commutative cuts

$$\begin{array}{lll}
(\zeta_{\vee}) & F[\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r]] & \xrightarrow{h} \; \mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to F[q] | \iota_2(b) \to F[r]] \\
(\zeta_{\exists}) & F[\mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q] & \xrightarrow{h} \; \mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ F[q] \\
(\zeta_{\perp}) & F[\mathsf{efq}(p)] & \xrightarrow{h} \; \mathsf{efq}(p)
\end{array}$$

The left-hand side of a rule is called *redex* and the right-hand side *contractum* or *reduct.*

We write $\overset{h}{\to}$ to mean one application of the rule at the head of the term and $\to$ for the congruent closure of $\overset{h}{\to}$ over all constructions. We write $\twoheadrightarrow$ for the reflexive-transitive closure of $\to$ and $=$ for the reflexive-symmetric-transitive closure. To refer to one of the rule specifically, e.g. $(\beta_\Rightarrow)$, one writes respectively $\overset{h}{\to}_{\beta_\Rightarrow}$, $\to_{\beta_\Rightarrow}$, $\twoheadrightarrow_{\beta_\Rightarrow}$ and $=_{\beta_\Rightarrow}$. To refer to all β-rules at once, one write $\overset{h}{\to}_\beta$, $\to_\beta$, $\twoheadrightarrow_\beta$ and $=_\beta$. Similarly to refer to all ζ-rules at once.

Remark 8 *The extra inference rules for positive conjunction, and positive true formula can be annotated using the extra expressions described by the following extended grammar:*

$$p, q \quad ::= \quad \ldots \mid (p,q)_+ \mid \mathsf{dest}\ p\ \mathsf{as}\ (a,b)_+\ \mathsf{in}\ q \mid ()_+ \mid \mathsf{dest}\ p\ \mathsf{as}\ ()_+\ \mathsf{in}\ q$$

$$\frac{\Gamma \vdash_I p : A \quad \Gamma \vdash_I q : B}{\Gamma \vdash_I (p,q)_+ : A \wedge B}\ \wedge'_I \qquad \frac{\Gamma \vdash_I p : A \wedge B \quad \Gamma, a : A, b : B \vdash_I q : C}{\Gamma \vdash_I \mathsf{dest}\ p\ \mathsf{as}\ (a,b)_+\ \mathsf{in}\ q : C}\ \wedge'_E$$

$$\frac{}{\Gamma \vdash_I ()_+ : \top}\ \top'_I \qquad \frac{\Gamma \vdash_I p : \top \qquad \Gamma \vdash_I q : C}{\Gamma \vdash_I \mathsf{dest}\ p\ \mathsf{as}\ ()_+\ \mathsf{in}\ qq : C}\ \top'_E$$

Remark 9 *To interpret the alternative presentation of intuitionistic logic with two kinds of sequent, we shall use a second syntactic categories of annotations called commands to interpret sequents of the form $\Gamma \vdash$. In this context, the alternative inference rules for the negative false formula can be annotated using syntax defined by the following grammar:*

$$\begin{array}{lll} p, q & ::= & \ldots \mid \mu\_.c \\ c & ::= & [tp]p \end{array}$$

We shall write $c : (\Gamma \vdash)$ for an annotated sequent of the second kind. The annotated inference rules are then:

$$\frac{\Gamma \vdash_I p : \bot}{[tp]p : (\Gamma \vdash_I)}\ \bot'_E \qquad \frac{c : (\Gamma \vdash_I)}{\Gamma \vdash_I \mu\_.c : A}\ weak_R$$

That these rules can be interpreted as proof transformations (via the Curry-Howard isomorphism) results from the following theorem of *preservation of typing*, also called *subject reduction.* Before proving the theorem, we however need a proposition stating stabilility by substitution.

Lemma 8 (Substitution) *If $\Gamma_1, a : A, \Gamma_2 \vdash_I p : B$ and $\Gamma_1, \Gamma_2 \vdash_I q : A$, then $\Gamma_1, \Gamma_2 \vdash_I p[q/a] : B$.*

Proof: By induction on p, reducing to the case of an axiom $\Gamma_1, a : A, \Gamma_2 \vdash_I a : A$. But $\Gamma_1, \Gamma_2 \vdash_I a[q/a] : A$ holds by hypothesis. Note that we use the Weakening Lemma to adjust the typing context of q for rules which add formulas to the context. ■

Theorem 1 (Subject reduction) *If $\Gamma \vdash_I p : A$ and $p \to q$ then $\Gamma \vdash_I q : A$.*

Proof: By induction on the definition of $p \to q$, reducing it to the base case $p \overset{h}{\to} q$. All cases follow by analysis of the possible forms of a derivation annotated with p and showing that we can build a proof of the same sequent annotated by q. We treat for instance the case of $(\beta_\Rightarrow)$ and $(\zeta_\vee)$.

If $\Gamma \vdash_I (\lambda a.p)q : A$, then, necessarily, $\Gamma, a : B \vdash p : A$ and $\Gamma \vdash q : B$ for some B. Then, $\Gamma \vdash_I p[q/a] : A$ by the Substitution Lemma.

If $\Gamma \vdash_I F[\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r]] : A$ then, necessarily, $\Gamma \vdash p : B \vee C$, $\Gamma, a : B \vdash q : D$, $\Gamma, b : C \vdash r : D$ and $\Gamma, d : D \vdash_I F[d] : A$ for some B, C and D. We have $\Gamma \vdash_I F[q] : A$ and $\Gamma \vdash_I F[r] : A$ by the Substitution Lemma, then, $\Gamma, a : B \vdash_I F[q] : A$ and $\Gamma, b : C \vdash_I F[r] : A$ by the Weakening Lemma. We conclude $\Gamma \vdash_I \mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to F[q] | \iota_2(b) \to F[r]] : A$ by elimination of disjunction. ■

A proof is said to be in *normal* form if no reduction rule is applicable. A key result about the above reduction system is that it is *normalising*, i.e. that for proofs, i.e. typed programs, a normal form is eventually obtained after applying reduction rules some finite number of times in some given order. The system is actually even *strongly normalising*, meaning that no matter which sequence of rules is applied at any place in the proof and in any order, a normal form is obtained.

Theorem 2 (Weak normalisation) *For any (annotated) proof $\Gamma \vdash_I p : A$, there exists an (annotated) proof $\Gamma \vdash_I q : A$ such that $p \twoheadrightarrow q$ and no reduction rules is applicable to q.*

Theorem 3 (Strong normalisation) *For any (annotated) proof $\Gamma \vdash_I p : A$, all sequences of reduction from p along $\to$ eventually reaches a normal form.*

Note that some of the rules are overlapping, e.g. $\zeta_\vee$ with $\beta^1_\vee$. Such a configuration is called a *critical pair*, but in all cases of overlapping, the critical pairs *converge*, in the sense that we can apply zero, one or more rules on each reduct so that the proofs become the same.

There are also cases of overlapping due to the use of substitution, e.g. when applying $\beta_\Rightarrow$ at two nested positions of a proof, and these cases are shown to be convergent too, using general results on rewriting systems with binders. See [Ter03] for a general reference on rewriting systems.

As a consequence, the reduction rules give the same result whatever reduction path is taken, what we express as a confluence theorem:

Theorem 4 (Confluence) *If $p \twoheadrightarrow q_1$ and $p \twoheadrightarrow q_2$, there exists r such that $q_1 \twoheadrightarrow r$ and $q_2 \twoheadrightarrow r$. In particular, if $p \twoheadrightarrow q$ and q is normal, then q is unique satisfying this property.*

A key corollary of the normalisation theorem is the *subformula property.*

We say that a formula is an *immediate subformula in the semantical sense* when it is an immediate subformula in the syntactic sense, i.e. one of A or B for $A \wedge B$, $A \vee B$ and $A \Rightarrow B$, as well as when it has the form $A[t/x]$ for $\forall x\, A$ and $\exists x\, A$ and t a term. To be a *subformula in the semantical sense* is then the reflexive-transitive closure of being an immediate subformula.

Theorem 5 (Subformula property) *If $\Gamma \vdash A$ holds, then there is a derivation of $\Gamma \vdash A$ which does not mention any formulae other than subformulae of A or subformulae of the formulae in Γ, in the semantical sense of subformulae.*

In particular, this theorem is a key ingredient for proof search as it restricts the search space to those derivations which only mention subformulae of the sequent to be proved.

3.3 Structure of normal forms

Thanks to the normalisation theorem, we can ensure that any provable statement has a proof which is unique by confluence. This proof is witnessed by a program in the language defined by the following grammar:

$$\begin{array}{lcl} n, m & ::= & h \mid () \mid (n, n) \mid \iota_1(n) \mid \iota_2(n) \mid \lambda a.n \mid \lambda x.n \mid (t, n) \\ h & ::= & s \mid \mathsf{efq}\, s \mid \mathsf{case}\ s\ \mathsf{of}\ [\iota_1(a) \to n | \iota_2(b) \to m] \mid \mathsf{dest}\ s\ \mathsf{as}\ (x, a)\ \mathsf{in}\ n \\ s & ::= & a \mid \pi_1(s) \mid \pi_2(s) \mid s\, n \mid s\, t \end{array}$$

Lemma 9 *If $\Gamma \vdash_I n : A$, then n is in normal form.*

Proof: By combinatoric inspection of the possible forms of n. ∎

Theorem 6 (Canonical form of proofs) *If $\Gamma \vdash_I A$, then there is n in the language of normal forms such that $\Gamma \vdash_I n : A$.*

3.4 Eta-expansion

We can constrain further the canonical forms by demanding that any h is of positive or atomic type and, if of positive type, an elimination of a positive connective. This is obtained by so-called *η-expansions*, which enter the category of observational rules, and which need types:

observational rules

$(\eta_\Rightarrow)$	p	$\xrightarrow{h}$	$\lambda a.(p\, a)$	if $\Gamma \vdash_I p : A \Rightarrow B$	
$(\eta_\wedge)$	p	$\xrightarrow{h}$	$(\pi_1(p), \pi_2(p))$	if $\Gamma \vdash_I p : A \wedge B$	(also called *surjective pairing*)
$(\eta_\vee)$	p	$\xrightarrow{h}$	$\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(b) \to \iota_1(b) \mid \iota_2(c) \to \iota_2(c)]$	if $\Gamma \vdash_I p : A \vee B$	
$(\eta_\top)$	p	$\xrightarrow{h}$	$()$	if $\Gamma \vdash_I p : \top$	
$(\eta_\bot)$	p	$\xrightarrow{h}$	$\mathsf{efq}\, p$	if $\Gamma \vdash_I p : \bot$	
$(\eta_\forall)$	p	$\xrightarrow{h}$	$\lambda x.(p\, x)$	if $\Gamma \vdash_I p : \forall x\, A$	
$(\eta_\exists)$	p	$\xrightarrow{h}$	$\mathsf{dest}\ p\ \mathsf{as}\ (x, b)\ \mathsf{in}\ (x, b)$	if $\Gamma \vdash_I p : \exists x\, A$	

One can check that η-expansion rules preserves typing as soon as p has the correct type.

The calculus has the same structural and defining rules as in Figure 2.2 with annotations as in Figure 3.1 and the two rules of Figure 2.2 which are not in Figure 3.1 annotated as follows:

extra structural rules

$$\frac{\Gamma \vdash_C \Delta_1, \alpha : A, \Delta_2; p : A}{[\alpha]p : (\Gamma \vdash_C \Delta_1, \alpha : A, \Delta_2;)}\ \textit{Unfocus} \qquad \frac{c : (\Gamma \vdash_C \Delta, \alpha : A;)}{\Gamma \vdash_C \Delta; \mu\alpha.c : A}\ \textit{Focus}$$

Figure 3.2: Classical natural deduction (Parigot's style), annotated

Using η-expansions on expressions of the form s leads then to enter the following refined grammar of normal forms:

$$\begin{array}{lcl} n, m & ::= & h \mid () \mid (n, n) \mid \iota_1(n) \mid \iota_2(n) \mid \lambda a.n \mid \lambda x.n \mid (t, n) \\ h & ::= & \mathsf{efq}\, s \mid \mathsf{case}\ s\ \mathsf{of}\ [\iota_1(a) \to n | \iota_2(b) \to m] \mid \mathsf{dest}\ s\ \mathsf{as}\ (x, a)\ \mathsf{in}\ n \\ s & ::= & a \mid \pi_1(s) \mid \pi_2(s) \mid s\, n \mid s\, t \end{array}$$

where terms covered by the syntactic entry h are additionally of a positive type.

Indeed, if some h is of a negative type, we apply one of $(\eta_\Rightarrow)$, $(\eta_\wedge)$, $(\eta_\top)$ or $(\eta_\forall)$ and get, after reduction of possible newly created commutative cuts, an expression involving new terms of smaller type in h. If some h is an s of positive type, we apply one of $(\eta_\vee)$, $(\eta_\bot)$ or $(\eta_\exists)$ and get an h which is not of the form s.

Note that η-rules applied to terms corresponding to proofs starting with an introduction rule, or to terms which are in the context of (the main position of) an elimination rule, introduce loops in the reduction system since then the application of the η rule can be reverted using a β-rule (e.g. $\lambda a.p \to_\eta \lambda b.((\lambda a.p)\, b) \to \lambda b.p[b/a] \equiv p$, or, $a\, p \to_\eta (\lambda b.(a\, b))\, p \to a\, p$).

So, for a rewriting system not to be looping, η-rules should be applied to terms corresponding to proofs starting with an elimination rule or an axiom and which are not themselves in the context of an elimination, what indeed essentially corresponds to being an s in the entry h.

3.5 The classical case

We now extend the annotation of proofs to the classical case. To each kind of sequent, we associate an annotation in a specific class of expressions. To annotate sequents of the form $\Gamma \vdash_C \Delta; A$, we use terms as defined for intuitionistic natural deduction but extended with the following μ-construction:

$$p \quad ::= \quad ... \mid \mu\alpha.c$$

To annotate sequents of the form $\Gamma \vdash_C \Delta;$, we use annotations called *commands*, ranged over by letter c and described by the following grammar:

$$c \quad ::= \quad [\alpha]p$$

The annotation is then written as follows: $c : (\Gamma \vdash_C \Delta;)$.

In $\mu\alpha.c$ and $[\alpha]p$, α is a name for a formula in Δ. The construction $\mu\alpha.c$ binds α and in the construction $[\alpha]p$ is bound by closest surrounding $\mu\alpha$. If no such binder is present, as in the general cases of binders, the variable is called free.

In the annotation of inference rules, the formulae in Δ come with a name ranged over by α, β, γ, ... so that the new definition of Δ is:

$$\Delta \quad ::= \quad \emptyset \mid \Delta, \alpha : A$$

The inference rules which only differ from those of intuitionistic logic by the presence of Δ are annotated by exactly the same program constructions as in the intuitionistic case. As for the inference rules specific to classical reasoning, they are annotated using notations introduced by Parigot [Par91] as shown in Figure 3.2.

3.6 Basic properties

Extending the annotated case to classical logic, we have the following extended weakening, contraction and strengthening lemmas. They apply not only to the left-hand side but also to the right-hand side of the sequents. They apply

as well for sequents with a focus and for sequents without a focus and the proofs are by mutual induction on these two kinds of sequents.

Lemma 10 (Admissibility of weakening)
If $\Gamma_1, \Gamma_2 \vdash p : A$ *and* b *is not already used in* Γ_1, Γ_2*, then* $\Gamma_1, b : B, \Gamma_2 \vdash \Delta; p : A$.
If $\Gamma_1, \Gamma_2 \vdash c$ *and* b *is not already used in* Γ_1, Γ_2*, then* $\Gamma_1, b : B, \Gamma_2 \vdash \Delta; c$.
If $\Gamma \vdash \Delta_1, \Delta_2; p : A$ *and* β *is not already used in* Δ_1, Δ_2*, then* $\Gamma \vdash \Delta_1, \beta : B, \Delta_2; p : A$.
If $\Gamma \vdash \Delta_1, \Delta_2; c$ *and* β *is not already used in* Δ_1, Δ_2*, then* $\Gamma \vdash \Delta_1, \beta : B, \Delta_2; c$.

Lemma 11 (Admissibility of contraction)
If $\Gamma_1, a_1 : A, \Gamma_2, a_2 : A, \Gamma_3 \vdash \Delta; p : B$ *then* $\Gamma_1, a_1 : A, \Gamma_2, \Gamma_3 \vdash p[a_1/a_2] : B$ *and* $\Gamma_1, \Gamma_2, a_2 : A, \Gamma_3 \vdash p[a_2/a_1] : B$.
If $\Gamma_1, a_1 : A, \Gamma_2, a_2 : A, \Gamma_3 \vdash \Delta; c$ *then* $\Gamma_1, a_1 : A, \Gamma_2, \Gamma_3 \vdash c[a_1/a_2]$ *and* $\Gamma_1, \Gamma_2, a_2 : A, \Gamma_3 \vdash c[a_2/a_1]$.
If $\Gamma \vdash \Delta_1, \beta_1 : B, \Delta_2, \beta_2 : B, \Delta_3; p : A$ *then* $\Gamma \vdash \Delta_1, \beta_1 : B, \Delta_2, \Delta_3; p[\beta_1/\beta_2] : A$ *and* $\Gamma \vdash \Delta_1, \Delta_2, \beta_2 : B, \Delta_3; p[\beta_2/\beta_1] : A$.
If $\Gamma \vdash \Delta_1, \beta_1 : B, \Delta_2, \beta_2 : B, \Delta_3; c$ *then* $\Gamma \vdash \Delta_1, \beta_1 : B, \Delta_2, \Delta_3; c[\beta_1/\beta_2]$ *and* $\Gamma \vdash \Delta_1, \Delta_2, \beta_2 : B, \Delta_3; c[\beta_2/\beta_1]$.

Lemma 12 (Admissibility of strengthening)
If $\Gamma_1, a : A, \Gamma_2 \vdash \Delta; p : B$ *and* $a \notin FV(p)$ *then* $\Gamma_1, \Gamma_2 \vdash \Delta; p : B$.
If $\Gamma_1, a : A, \Gamma_2 \vdash \Delta; c$ *and* $a \notin FV(c)$ *then* $\Gamma_1, \Gamma_2 \vdash \Delta; c$.
If $\Gamma \vdash \Delta_1, \alpha : A, \Delta_2; p : B$ *and* $\alpha \notin FV(p)$ *then* $\Gamma \vdash \Delta_1, \Delta_2; p : B$.
If $\Gamma \vdash \Delta_1, \alpha : A, \Delta_2; c$ *and* $\alpha \notin FV(c)$ *then* $\Gamma \vdash \Delta_1, \Delta_2; c$.

3.7 Reduction rules

Normalisation of proofs is obtained by adding one reduction rule taking the form of a commutative cut rule and one reduction rule for collapsing μ-binders. Altogether, the rules can be seen as rules for incrementally moving the elementary components of the evaluation context surrounding a μ-expression until this context is a $[\beta]$.

incremental context substitution

$$\begin{array}{lcl} (\zeta_\mu) & F[\mu\alpha.c] & \stackrel{h}{\to} \quad \mu\beta.c[[\beta]F/\alpha] \\ (\zeta_c) & [\beta]\mu\alpha.c & \stackrel{h}{\to} \quad c[\beta/\alpha] \end{array}$$

where the notation $c[[\alpha]F/\alpha]$, called *Parigot's structural substitution*, denotes the replacement of any subcommand of the form $[\alpha]p$ with the command $[\alpha](F[p])$, and recursively. Otherwise said, this is a substitution of elementary evaluation contexts. Formally, its definition requires also the definition of $p[[\alpha]F/\alpha]$, and it is obtained by:

$$\begin{array}{lcl} ([\alpha]p)[[\alpha]F/\alpha] & \triangleq & [\alpha](F[p[[\alpha]F/\alpha]]) \\ a[[\alpha]F/\alpha] & \triangleq & a \\ (p,q)[[\alpha]F/\alpha] & \triangleq & (p[[\alpha]F/\alpha], q)[[\alpha]F/\alpha]) \\ \pi_1(p)[[\alpha]F/\alpha] & \triangleq & \pi_1(p[[\alpha]F/\alpha] \\ \pi_2(p)[[\alpha]F/\alpha] & \triangleq & \pi_2(p[[\alpha]F/\alpha] \\ \text{etc.} & & \end{array}$$

One may find alternative notations for $p[[\beta]F/\alpha]$ in the literature. For instance, Parigot's own original notation, which was specialised to the case of the context $[\,]\,q$ was written $p[[\alpha](r\,q)/[\alpha]r]$ which we could better write $p[[\alpha](\bullet\, q)/[\alpha]\bullet]$ to avoid the confusion on r behaving as a binder, and generalised to arbitrary contexts as $p[[\alpha](F[\bullet])/[\alpha]\bullet]$. Note that the notation $c[\beta/\alpha]$ could be itself rephrased using the notation of structural substitution. Indeed, it is equivalent to $c[[\beta][\,]/\alpha]$ (i.e. $c[[\beta]\bullet/[\alpha]\bullet]$).

Sometimes, as in Parigot's original notation above, one also reuses the same variable name in the right-hand side of the rules, writing instead $\mu\alpha.c[[\alpha]F/\alpha]$ which differs from $\mu\beta.c[[\beta]F/\alpha]$ only by an α-conversion, but a renaming which can sometimes be source of confusion, and which we therefore avoid.

A simplification rule could be considered too:

simplification rule

$$(\Omega_\mu) \quad \mu\alpha.[\alpha]p \quad \stackrel{h}{\to} \quad p \quad \text{if } \alpha \text{ not free in } p$$

Elementary evaluation contexts can be nested, and even combined with a leading $[\alpha]$, leading to the following grammar of (non necessarily elementary) evaluation contexts, ranged over by the letter E, and of jumps ranged over by the letter J:

call-by-name evaluation contexts	E	::=	$[\,] \mid E[F]$
call-by-name jumps	J	::=	$[\alpha]E$

where we can now interpret $E[p]$ as the macro-notation for the term defined by the clauses $([\,])[p] \triangleq p$ and $(E[F])[p] \triangleq E[F[p]]$, and similarly for $J[p]$.

The syntax of formulae, contexts and proofs, as well as the typing system and the reduction rules completely define *call-by-name classical natural deduction.*

We have the same properties as in the intuitionistic case. To start with, Substitution has now new cases. New cases because of the distinction between sequents $\Gamma \vdash \Delta; p : A$ and $\Gamma \vdash \Delta; c$, and new cases because of substitution of evaluation contexts.

Lemma 13 (Substitution) *If* $\Gamma_1, a : A, \Gamma_2 \vdash_C \Delta; p : B$ *and* $\Gamma_1, \Gamma_2 \vdash_C \Delta; q : A$, *then* $\Gamma_1, \Gamma_2 \vdash_C \Delta; p[q/a] : B$.
If $\Gamma_1, a : A, \Gamma_2 \vdash_C \Delta; c$ *and* $\Gamma_1, \Gamma_2 \vdash_C \Delta; q : A$, *then* $\Gamma_1, \Gamma_2 \vdash_C \Delta; c[q/a]$.
If $\Gamma \vdash_C \Delta_1, \alpha : A, \Delta_2; p : B$ *and* $\Gamma, a : A \vdash_C \Delta_1, \Delta_2; [\beta](F[a])$, *then* $\Gamma \vdash_C \Delta_1, \Delta_2; p[[\beta]F[\,]/\alpha] : B$.
If $\Gamma \vdash_C \Delta_1, \alpha : A, \Delta_2; c$ *and* $\Gamma, a : A \vdash_C \Delta_1, \Delta_2; [\beta](F[a])$, *then* $\Gamma \vdash_C \Delta_1, \Delta_2; c[[\beta]F[\,]/\alpha]$.

Proof: The first two cases are as in the intuitionistic case, using a mutual induction on the derivation of p and c, using weakening for rules which extend the context. For the last two cases, this is also a mutual induction on the derivation of p and c, also using for rules which extend the context. This eventually reduces to the case $\Gamma \vdash_C \Delta_1, \alpha : A, \Delta_2; [\alpha]r$ which can be composed with $\Gamma, a : A \vdash_C \Delta_1, \Delta_2; [\beta](F[a])$ to get $\Gamma \vdash_C \Delta_1, \Delta_2; [\beta](F[r])$. ■

Substitution allows to prove the preservation of typing by reduction.

Theorem 7 (Preservation of typing by reduction (Subject reduction)) *If* $\Gamma \vdash_C \Delta; p : A$ *and* $p \to q$ *then* $\Gamma \vdash_C \Delta; q : A$. *If* $c : (\Gamma \vdash_C \Delta)$ *and* $c \to c'$ *then* $c' : (\Gamma \vdash_C \Delta)$.

Proof: By reasoning by induction on the mutual definition of $p \to q$ and $c \to c'$, this reduces to the cases $p \xrightarrow{h} q$ and $c \xrightarrow{h} c'$. For instance, we use Substitution of evaluation contexts to deal with the new rule (ζ_μ). ■

The notion of normal proof, normalising reduction system, strongly normalising reduction system applies to the extended reduction system. The following properties hold.

Theorem 8 (Weak normalisation) *For any (annotated) proof* $\Gamma \vdash_C \Delta; p : A$, *there exists an (annotated) proof* $\Gamma \vdash_C \Delta; q : A$ *such that* $p \twoheadrightarrow q$ *and no reduction rules is applicable to* q.

Theorem 9 (Strong normalisation) *For any (annotated) proof* $\Gamma \vdash_I p : A$, *the number of steps of all paths of reduction from* $p \twoheadrightarrow q$ *for any* q *is bounded.*

Confluence holds but its proof is a bit more involved.

Theorem 10 (Confluence) *If* $p \twoheadrightarrow q_1$ *and* $p \twoheadrightarrow q_2$, *there exists* r *such that* $q_1 \twoheadrightarrow r$ *and* $q_2 \twoheadrightarrow r$. *In particular, if* $p \twoheadrightarrow q$ *and* q *is normal, then* q *is unique satisfying this property.*

Proof: See Baba *al* [BHF01]. ■

From normalisation, we again get the subformula property.

Theorem 11 (Subformula property) *If* $\Gamma \vdash A$ *holds, then there is a derivation of* $\Gamma \vdash A$ *which does not mention any other formulae than subformulae of* A *or subformulae of the formulae in* Γ, *in the semantical sense of subformulae.*

3.8 Call-by-value semantics

3.9 Control operators in λ-calculus

Chapter 4

The proofs-as-programs interpretation of Frege-Hilbert's systems

Predicate calculus as a formal system of proofs was invented by Frege [Fre67] and the kind of system he designed is nowadays known as Frege-Hilbert system or Hilbert system. A characteristic feature of this kind of systems is that when deriving a judgement of the form $\Gamma \vdash A$, the context Γ is unchanged throughout the derivation. Such system is built from axioms and two inference rules, modus ponens and generalisation. The computational counterpart to such system is Schönfinkel's combinatory logic as was noticed by Curry [Cur34].

Frege's original system used only implication, negation and universal quantification as primitive connectives and quantifiers. There are several Frege-Hilbert-style presentations of predicate logic in the literature (see e.g.[**?**]). We shall give one with all five standard connectives and quantifiers, plus the always true and always false connectives.

We directly present the system with a proofs-as-programs presentation. We can recover a standard formulation of a Frege-Hilbert's system by dropping the proof annotations.

4.1 The intuitionistic case

We annotate sequents by programs defined by the following grammar, which follows the structure of proofs:

$$p, q, r \quad ::= \quad a \mid p\,q \mid K \mid S \mid 1 \mid 0 \mid P \mid P_1 \mid P_2 \mid Q_1 \mid Q_2 \mid Q \mid F_t \mid C_\forall \mid \lambda x.p \mid E_t \mid E$$

These programs are actually programs of combinatory logic. So, in this section, $\Gamma \vdash_I p : A$ can alternatively be read as p is an intuitionistic proof of $\Gamma \vdash_I A$ or p is program of combinatory logic typed by A in environment Γ according to the system of simple types corresponding to predicate logic.

$$\frac{}{\Gamma_1, a : A, \Gamma_2 \vdash_I a : A}\ Ax \qquad \frac{}{\Gamma \vdash_I 1 : \top}\ Ax^\top \qquad \frac{}{\Gamma \vdash_I 0 : \bot \Rightarrow A}\ Ax^\bot$$

$$\frac{}{\Gamma \vdash_I P : A \Rightarrow B \Rightarrow A \wedge B}\ Ax^\wedge_I \qquad \frac{}{\Gamma \vdash_I P_1 : A \wedge B \Rightarrow A}\ Ax^\wedge_{E^1} \qquad \frac{}{\Gamma \vdash_I P_2 : A \wedge B \Rightarrow B}\ Ax^\wedge_{E^2}$$

$$\frac{}{\Gamma \vdash_I Q_1 : A \Rightarrow A \vee B}\ Ax^\vee_{I^2} \qquad \frac{}{\Gamma \vdash_I Q_2 : B \Rightarrow A \vee B}\ Ax^\vee_{I^2} \qquad \frac{}{\Gamma \vdash_I Q : (A \Rightarrow C) \Rightarrow (b \Rightarrow C) \Rightarrow A \vee B \Rightarrow C}\ Ax^\vee_E$$

$$\frac{}{\Gamma \vdash_I K : A \Rightarrow B \Rightarrow A}\ Ax_K \qquad \frac{}{\Gamma \vdash_I S : (A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow (A \Rightarrow C)}\ Ax_S \qquad \frac{\Gamma \vdash_I p : A \Rightarrow B \quad \Gamma \vdash_I q : A}{\Gamma \vdash_I p\,q : B}\ MP$$

$$\frac{}{\Gamma \vdash_I F_t : \forall x\, A \Rightarrow A[t/x]}\ Ax^\forall \qquad \frac{x \text{ not occurring free in } A}{\Gamma \vdash_I C_\forall : \forall x\,(A \Rightarrow B) \Rightarrow A \Rightarrow \forall x\, B}\ Ax_{C_\forall} \qquad \frac{\Gamma \vdash_H p : A \quad x \text{ not occurring free in } \Gamma}{\Gamma \vdash_I \lambda x.p : \forall x\, A}\ Gen_\forall$$

$$\frac{}{\Gamma \vdash_I E_t : A[t/x] \Rightarrow \exists x\, A}\ Ax^\exists_I \qquad \frac{x \text{ not occurring free in } C}{\Gamma \vdash_I E : \forall x\,(A \Rightarrow C) \Rightarrow \exists x\, A \Rightarrow C}\ Ax^\exists_E$$

Frege-Hilbert's systems can be equipped with a reduction system. The two first rules are standard and mimic the computational behaviour of combinatory logic. The other rules are specific to our presentation.

$$\begin{array}{lcl}
K\,p\,q & \stackrel{h}{\to} & p \\
S\,p\,q\,r & \stackrel{h}{\to} & p\,r\,(q\,r) \\
P_1\,(P\,p\,q) & \stackrel{h}{\to} & p \\
P_2\,(P\,p\,q) & \stackrel{h}{\to} & q \\
Q\,q\,r\,(Q_1\,p) & \stackrel{h}{\to} & q\,p \\
Q\,q\,r\,(Q_2\,p) & \stackrel{h}{\to} & r\,p \\
E\,q\,(E_t\,p) & \stackrel{h}{\to} & F_t\,q\,p \\
F_t\,(\lambda x.p) & \stackrel{h}{\to} & p[t/x] \\
C_\forall\,(\lambda x.p)\,q & \stackrel{h}{\to} & \lambda x.(p\,q)
\end{array}$$

This system is interdefinable with natural deduction but while the reduction of this Frege-Hilbert's system can be simulated within natural deduction, the converse is not true (see e.g. [SU99] for details).

In particular, the reduction system of a Frege-Hilbert system is not fine-grained enough to ensure the subformula property. For instance, the proof $S\,K\,K$ of $A \Rightarrow A$ has subterms whose type is not a subtype of $A \Rightarrow A$ (consider e.g. the type $A \Rightarrow (A \Rightarrow A) \Rightarrow A$ of the first occurrence of K).

Remark 10 *We could also add commutative rules, but since the subformula property is anyway not guarantee, this is less interesting. The following are well-typed commutative rules:*

$$\begin{array}{lcl}
Q\,q\,r\,p\,p' & \stackrel{h}{\to} & Q\,(q\,p')\,(r\,p')\,p \\
E\,q\,p\,p' & \stackrel{h}{\to} & E\,(q\,p')\,p
\end{array}$$

A standard theorem of Frege-Hilbert's systems is the *deduction theorem*, which can be seen as the admissibility of the right introduction rule of implication.

Theorem 12 (Deduction theorem) *If* $\Gamma, A \vdash B$ *then* $\Gamma \vdash A \Rightarrow B$

This theorem has a computational counterpart in combinatory logic which is *bracket abstraction*, i.e. the definability of abstraction in combinatory logic (it is also called *combinatory abstraction* and sometimes written $\lambda^\star a.p$). The bracket abstraction of p with respect to some variable a is defined by the following clauses:

$$\begin{array}{lcll}
[a]a & \triangleq & S\,K\,K & \\
[a]p & \triangleq & K\,p & p \text{ a variable distinct from } a \text{ or a primitive combinator} \\
[a](p\,q) & \triangleq & S\,[a]p\,[a]q & \\
[a](\lambda x.p) & \triangleq & C_\forall\,\lambda x.[a]p &
\end{array}$$

The following is a computationally more informative formulation of the previous theorem:

Theorem 13 (Deduction theorem, computational form) *If* $\Gamma, a : A \vdash p : B$ *then* $\Gamma \vdash [a]p : A \Rightarrow B$

Proof: By induction on p verifying that each step of the construction of $[a]p$ preserves typing. ■

4.2 The classical case

The classical case can be obtained in different ways. In the axiomatic spirit of Frege-Hilbert's systems, we shall obtain classical logic by adding an axiom for Peirce's law.

$$p, q, r \quad ::= \quad \ldots \mid L$$

The corresponding axiom extends the previous system $\vdash_I$ into a classical system $\vdash_C$.

$$\frac{}{\Gamma \vdash_C L : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}\ Ax_L$$

Exercice 2 *Give reduction rules to* L *by simulating the natural deduction case.*

Chapter 5

Negative translations and continuation-passing-style translations

After the informal ideas underlying intuitionistic logic were developed by Brouwer, formal systems for respectively intuitionistic and minimal provability were given by Heyting and Kolmogorov. Heyting simply removed the axiom of excluded-middle leading to a system with same provability as intuitionistic natural deduction. Kolmogorov additionally removed *ex falso quodlibet*, i.e. the schema $\bot \Rightarrow A$, also called *principle of explosion*, leading to minimal logic. Kolmogorov [Kol67] also showed that classical logic can be embedded into minimal logic by inserting a certain amount of double-negation. This was later rediscovered by Gödel [Göd32] and Gentzen [Gen36], with several variants around, especially one by Kuroda [Kur51], one for negative intuitionistic logic by Krivine [Kri90].

The programming counterpart to negative translations are continuation-passing-style translations, as shown by Murthy [Mur90] shortly after Griffin established a connection between control operators and classical reasoning [Gri90].

Continuation-passing-style (shortly cps) translations were introduced in the 60's to simulate `goto`'s in a purely functional way, in a style where functions never return but instead eventually call a continuation passed as an argument of the function and which represents the rest of the computation to do(see Reynolds [Rey93] for an history of the discovery of continuations). Continuation-passing-style translations were also used to force the evaluation order of a language. This was in particular the case of Fischer's and Plotkin's cps-translations of call-by-value λ-calculus into a fragment of λ-calculus insensitive to the evaluation order [Fis72, Plo75]. This was also the case of Plotkin's cps-translation of call-by-name λ-calculus into this same fragment insensitive to the evaluation order [Plo75]. Abstraction of `goto` where proposed by Landin, with his operator J [Lan65], Reynolds, with his operator `escape` [Rey72], Scheme with its `catch`/`throw` constructs [SS75] which evolved into a stand-alone operator named `call-with-current-continuation`, shortly `call/cc`, in 1982 [KCR$^+$98]. That `call/cc` is interpretable by continuation-passing-style can be found in Clinger, Friedman and Wand [WC85]. A new post-Griffin optimised continuation-passing-style was further designed for call-by-name by Lafont-Reus-Streicher [LRS93]. We shall present these translations, adapted to first-order predicate logic.

5.1 Kolmogorov's translation and call-by-name evaluation

Kolmogorov's translation inserts a double negation in front of any formula. We define it in two steps: a translation A^+_{Kol} which inserts a double negation in front of every subformulae and a translation A^*_{Kol} which inserts an extra double negation at the head of the formula.

$$
\begin{array}{lcl}
A^*_{Kol} & \triangleq & \neg\neg A^+_{Kol} \\
 & & \\
X(t_1,..,t_n)^+_{Kol} & \triangleq & X(t_1,...,t_n) \\
\bot^+_{Kol} & \triangleq & \bot \\
\top^+_{Kol} & \triangleq & \top \\
(A \wedge B)^+_{Kol} & \triangleq & A^*_{Kol} \wedge B^*_{Kol} \\
(A \vee B)^+_{Kol} & \triangleq & A^*_{Kol} \vee B^*_{Kol} \\
(A \Rightarrow B)^+_{Kol} & \triangleq & A^*_{Kol} \Rightarrow B^*_{Kol} \\
(\exists x\, A)^+_{Kol} & \triangleq & \exists x\, A^*_{Kol} \\
(\forall x\, A)^+_{Kol} & \triangleq & \forall x\, A^*_{Kol}
\end{array}
$$

The main point of Kolmogorov's translation is to obtain the following result:

Theorem 14 (Kolmogorov's embedding of classical logic in minimal logic)
$\Gamma \vdash_C \Delta; A$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M A^*_{Kol}$
$\Gamma \vdash_C \Delta;$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M \bot$

PROOF: The proof is by mutual induction on the derivation, showing that each inference rule holds through the translation. The interesting cases are the *Unfocus* and *Focus* rules. ■

5.1.1 Kolmogorov's translation as a continuation-passing-style translation

However, we can also be interested in the proof of the theorem as a program transformation, which is called *continuation-passing-style translation* in the context of programming.

Let us partition term variables into two countable subsets, one for interpreting the α variables (and we then write k_α for the bijection of the α variables into this subset), with the second subset used for interpreting the term variables. The following continuation-passing-style translation was defined by Plotkin's call-by-name translation [Plo75] to full first-order predicate logic:

$$
\begin{array}{ll}
a^*_{Kol} & \triangleq \ a \\
()^*_{Kol} & \triangleq \ \lambda k.k() \\
(\mathsf{efq}\, p)^*_{Kol} & \triangleq \ p(\lambda a.a) \\
(p,q)^*_{Kol} & \triangleq \ \lambda k.k(p^*_{Kol}, q^*_{Kol}) \\
\pi_1(p)^*_{Kol} & \triangleq \ \lambda k.p^*_{Kol}(\lambda a.\pi_1(a)k) \\
\pi_2(p)^*_{Kol} & \triangleq \ \lambda k.p^*_{Kol}(\lambda a.\pi_2(a)k) \\
\iota_1(p)^*_{Kol} & \triangleq \ \lambda k.k(\iota_1(p^*_{Kol})) \\
\iota_2(p)^*_{Kol} & \triangleq \ \lambda k.k(\iota_2(p^*_{Kol})) \\
\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a_1) \to q_1 | \iota_2(a_2) \to q_2]^*_{Kol} & \triangleq \ \lambda k.p^*_{Kol}(\lambda b.\mathsf{case}\ b\ \mathsf{of}\ [\iota_1(a_1) \to {q_1}^*_{Kol}k | \iota_2(a_2) \to {q_2}^*_{Kol}k]) \\
(\lambda a.p)^*_{Kol} & \triangleq \ \lambda k.k(\lambda a.p^*_{Kol}) \\
(p\,q)^*_{Kol} & \triangleq \ \lambda k.p^*_{Kol}(\lambda a.(a\, q^*_{Kol}\, k)) \\
(\lambda x.p)^*_{Kol} & \triangleq \ \lambda k.k(\lambda x.p^*_{Kol}) \\
(p\,t)^*_{Kol} & \triangleq \ \lambda k.p^*_{Kol}(\lambda a.(a\, t\, k)) \\
(t,p)^*_{Kol} & \triangleq \ \lambda k.k(t, p^*_{Kol}) \\
(\mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q)^*_{Kol} & \triangleq \ \lambda k.p^*_{Kol}(\lambda b.\mathsf{dest}\ b\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q^*_{Kol}k) \\
(\mu\alpha.c)^*_{Kol} & \triangleq \ \lambda k_\alpha.c^*_{Kol} \\
([\alpha]p)^*_{Kol} & \triangleq \ p^*_{Kol}k_\alpha
\end{array}
$$

We can now check that the above translation implements Theorem 14.

Theorem 15 (Kolmogorov's embedding of classical logic in minimal logic)
$\Gamma \vdash_C \Delta; p : A$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M p^*_{Kol} : A^*_{Kol}$
$c : (\Gamma \vdash_C \Delta;)$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M c^*_{Kol} : \bot$

5.1.2 The monad operators underlying Kolmogorov's continuation-passing-style translation

It happens that the translation can be reformulated on its minimal fragment using exclusively a type constructor $T(A)$ and two primitives traditionally written η and *bind* (or *), in reference to the property of $\neg\neg A$ to form a monad [Mog88].

$$
\begin{array}{lcll}
T(A) & \triangleq & \neg\neg A & \\
\eta & : & A \Rightarrow T(A) & \triangleq \ p \xrightarrow{h} \lambda k.(kp) \\
bind & : & (A \Rightarrow T(B)) \Rightarrow T(A) \Rightarrow T(B) & \triangleq \ q \mapsto p \mapsto \lambda k.(p\lambda a.(qak))
\end{array}
$$

We shall later use the notation $\mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ q$ for $bind\ (\lambda a.q)\, p$. The two operations satisfy the following properties:

$$
\begin{array}{lll}
\mathsf{bind}\ \eta(p)\ \mathsf{as}\ a\ \mathsf{in}\ q & \to_{\beta\eta} & qp \\
\mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ \eta\, a & \to_{\beta\eta} & p \\
\mathsf{bind}\ (\mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ q)\ \mathsf{as}\ b\ \mathsf{in}\ r & =_\beta & \mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ \mathsf{bind}\ q\ \mathsf{as}\ b\ \mathsf{in}\ r
\end{array}
$$

where in the last case, both sides converge toward a common proof.

The reformulation of the translation now emphasises its regularity, whether it interprets a right introduction rule or an elimination rule.

$$
\begin{array}{ll}
a^*_{Kol'} & \triangleq\ a \\
()^*_{Kol'} & \triangleq\ \eta\,() \\
(\mathsf{efq}\,p)^*_{Kol'} & \triangleq\ \mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ \lambda k.a \\
(p,q)^*_{Kol'} & \triangleq\ \eta(p^*_{Kol'}, q^*_{Kol'}) \\
\pi_1(p)^*_{Kol'} & \triangleq\ \mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ \pi_1(a) \\
\pi_2(p)^*_{Kol'} & \triangleq\ \mathsf{bind}\ p\ \mathsf{as}\ a\ \mathsf{in}\ \pi_2(a) \\
\iota_1(p)^*_{Kol'} & \triangleq\ \eta(\iota_1(p^*_{Kol'})) \\
\iota_2(p)^*_{Kol'} & \triangleq\ \eta(\iota_2(p^*_{Kol'})) \\
\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a_1) \to q_1 | \iota_2(a_2) \to q_2]^*_{Kol'} & \triangleq\ \mathsf{bind}\ p^*_{Kol'}\ \mathsf{as}\ b\ \mathsf{in}\ \mathsf{case}\ b\ \mathsf{of}\ [\iota_1(a_1) \to {q_1}^*_{Kol'} | \iota_2(a_2) \to {q_2}^*_{Kol'}] \\
(\lambda a.p)^*_{Kol'} & \triangleq\ \eta(\lambda a.p^*_{Kol'}) \\
(p\,q)^*_{Kol'} & \triangleq\ \mathsf{bind}\ p^*_{Kol'}\ \mathsf{as}\ a\ \mathsf{in}\ a\,q^*_{Kol'} \\
(\lambda x.p)^*_{Kol'} & \triangleq\ \eta(\lambda x.p^*_{Kol'}) \\
(p\,t)^*_{Kol'} & \triangleq\ \mathsf{bind}\ p^*_{Kol'}\ \mathsf{as}\ a\ \mathsf{in}\ a\,t \\
(t,p)^*_{Kol'} & \triangleq\ \eta(t, p^*_{Kol'}) \\
(\mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q)^*_{Kol'} & \triangleq\ \mathsf{bind}\ p^*_{Kol'}\ \mathsf{as}\ b\ \mathsf{in}\ \mathsf{dest}\ b\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q^*_{Kol'} \\
(\mu\alpha.c)^*_{Kol'} & \triangleq\ \lambda k_\alpha.c^*_{Kol'} \\
([\alpha]p)^*_{Kol'} & \triangleq\ p^*_{Kol'}\,k_\alpha
\end{array}
$$

Proposition 1 *We have* $p^*_{Kol'} =_\beta p^*_{Kol}$

And hence:

Theorem 16 (Kolmogorov's embedding of classical logic in minimal logic)
$\Gamma \vdash_C \Delta; p : A$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M \lambda k.p^*_{Kol'}k : A^*_{Kol}$
$c : (\Gamma \vdash_C \Delta;)$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M c^*_{Kol'} : \bot$

5.1.3 Kolmogorov's continuation-passing-style translation as a simulation

One might also be interested in showing that the translation simulates the reduction in natural deduction, but for that, one needs to optimise the translation. Indeed, consider for instance the translation of $p\,q\,r$. It is $\lambda k.((\lambda k'.p^*_{Kol}(\lambda f.(f\,q^*_{Kol}\,k')))\,(\lambda f'$ which shows a β-redex commonly called *administrative redex*. This redex contracts to $\lambda k.(p^*_{Kol}(\lambda f.(f\,q^*_{Kol}\,\lambda f'.(f'\,r^*_{Kol}\,k))))$ and the point of the translation below, sometimes referred to as Plotkin's "colon" translation, is to avoid the introduction of such administrative redexes by passing the current continuation K instead of abstracting over it at each step.

$$
\begin{array}{ll}
a^+_{Kol}K & \triangleq\ aK \\
()^+_{Kol}K & \triangleq\ K() \\
(\mathsf{efq}\,p)^+_{Kol}K & \triangleq\ p\lambda a.a \\
(p,q)^+_{Kol}K & \triangleq\ K(p^+_{Kol}, q^+_{Kol}) \\
\pi_1(p)^+_{Kol}K & \triangleq\ p^+_{Kol}(\lambda a.\pi_1(a)K) \\
\pi_2(p)^+_{Kol}K & \triangleq\ p^+_{Kol}(\lambda a.\pi_2(a)K) \\
\iota_1(p)^+_{Kol}K & \triangleq\ K(\iota_1(p^+_{Kol})) \\
\iota_2(p)^+_{Kol}K & \triangleq\ K(\iota_2(p^+_{Kol})) \\
(\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a_1) \to q_1 | \iota_2(a_2) \to q_2])^+_{Kol}K & \triangleq\ p^+_{Kol}(\lambda b.\mathsf{case}\ b\ \mathsf{of}\ [\iota_1(a_1) \to {q_1}^+_{Kol}K | \iota_2(a_2) \to {q_2}^+_{Kol}K]) \\
(\lambda a.p)^+_{Kol}K & \triangleq\ K(\lambda a.\lambda k.p^+_{Kol}) \\
(p\,q)^+_{Kol}K & \triangleq\ p^+_{Kol}(\lambda a.(a(\lambda k'.q^+_{Kol}k')K)) \\
(\lambda x.p)^+_{Kol}K & \triangleq\ K(\lambda x.\lambda k.p^+_{Kol}) \\
(p\,t)^+_{Kol}K & \triangleq\ p^+_{Kol}(\lambda a.(atK)) \\
(t,p)^+_{Kol}K & \triangleq\ K(t, p^+_{Kol}) \\
(\mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q)^+_{Kol}K & \triangleq\ p^+_{Kol}(\lambda b.\mathsf{dest}\ b\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q^+_{Kol}K) \\
(\mu\alpha.c)^+_{Kol}K & \triangleq\ (\lambda k_\alpha.c^+_{Kol})K \\
([\alpha]p)^+_{Kol} & \triangleq\ p^+_{Kol}k_\alpha
\end{array}
$$

Again, we have:

Theorem 17 (Kolmogorov's embedding of classical logic in minimal logic)
$\Gamma \vdash_C \Delta; p : A$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M \lambda k.p^+_{Kol}k : A^*_{Kol}$
$c : (\Gamma \vdash_C \Delta;)$ *implies* $\Gamma^*_{Kol}, \neg\Delta^+_{Kol} \vdash_M c^+_{Kol} : \bot$

The point of the modified translation is to obtain the following theorem:

Theorem 18 (Simulation of β-reduction) *If $p \to_\beta q$ then, for any k, $p^+_{Kol}k \twoheadrightarrow q^+_{Kol}k$*

One would expect to have the ζ-reduction to be simulated too, however it happens that through the translation, blocks of elementary evaluation contexts are captured at the same time, so that what can be simulated is only the following global ζ-reduction rule:

$$\begin{array}{lll}
(\zeta_\vee) & E[\mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to q | \iota_2(b) \to r]] & \stackrel{h}{\to}\ \mathsf{case}\ p\ \mathsf{of}\ [\iota_1(a) \to E[q] | \iota_2(b) \to E[r]] \\
(\zeta_\exists) & E[\mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ q] & \stackrel{h}{\to}\ \mathsf{dest}\ p\ \mathsf{as}\ (x,a)\ \mathsf{in}\ E[q] \\
(\zeta_\bot) & E[\mathsf{efq}(p)] & \stackrel{h}{\to}\ \mathsf{efq}(p)
\end{array}$$

5.2 Kuroda's translation and call-by-value evaluation

We now show a version of Kuroda's translation slightly optimised and modified so that it targets minimal logic: it highlights that we need double negations only at the head of positive connectives, at toplevel or within the scope of the negative connectives $\forall$ and $\Rightarrow$:

$$\begin{array}{lcl}
X(t_1,..,t_n)^* & \triangleq & \neg\neg X(t_1,...,t_n) \\
A \wedge B^* & \triangleq & A^* \wedge B^* \\
A \vee B^* & \triangleq & \neg\neg(A^+ \vee B^+) \\
(A \Rightarrow B)^* & \triangleq & A^+ \Rightarrow B^* \\
(\exists x\, A)^* & \triangleq & \neg\neg\exists x\, A^+ \\
(\forall x\, A)^* & \triangleq & \forall x\, A^*
\end{array}
\qquad
\begin{array}{lcl}
X(t_1,..,t_n)^+ & \triangleq & X(t_1,...,t_n) \\
A \wedge B^+ & \triangleq & A^+ \wedge B^+ \\
A \vee B^+ & \triangleq & A^+ \vee B^+ \\
(A \Rightarrow B)^+ & \triangleq & A^+ \Rightarrow B^* \\
(\exists x\, A)^+ & \triangleq & \exists x\, A^+ \\
(\forall x\, A)^+ & \triangleq & \forall x\, A^*
\end{array}$$

Theorem 19
$\Gamma \vdash_C \Delta; A$ *implies* $\Gamma^*, \neg\Delta^* \vdash_M A^*$
$\Gamma \vdash_C \Delta;$ *implies* $\Gamma^*, \neg\Delta^* \vdash_M A^*$

5.3 Lafont-Reus-Streicher's translation and call-by-name evaluation with η

5.4 Exercises

Exercice 3 *Show that in minimal logic, i.e. without assuming $\bot \Rightarrow A$ (Ex Falso Quodlibet, EFQ) to be granted, we have the following equivalences between schemes:*

$$DN \iff (PL + EFQ) \iff (EM + EFQ)$$

Show that we have also the following chain of implications between schemes:

$$DN \Rightarrow PL \Rightarrow EM$$

Exercice 4 *Show that in intuitionistic logic, i.e. in the presence of the scheme $\bot \Rightarrow A$, the three following schemes are equivalent:*

$A \vee \neg A$	*Excluded-middle (EM)*
$\neg\neg A \Rightarrow A$	*Double-negation elimination (DNE)*
$((A \Rightarrow B) \Rightarrow A) \Rightarrow A$	*Peirce's law (PL)*

In particular, this means that each of the three above axioms axiomatise classical logic on top of intuitionistic logic.

Exercice 5 *Show that $\Gamma, \neg\Delta, PL \vdash_I A$ iff $\Gamma \vdash_C \Delta; A$, where by $\Gamma, \neg\Delta, PL$ is meant the infinite context extending $\Gamma, \neg\Delta$ with all instances of the scheme PL.*

Exercice 6 *Show that there are two normal asymmetrical proofs of $\neg\neg A \wedge \neg\neg B \Rightarrow \neg\neg(A \wedge B)$ which are symmetric the one to the other.*

How many distinct proofs of $\neg\neg A \wedge \neg\neg B \wedge \neg\neg B \Rightarrow \neg\neg(A \wedge B \wedge C)$ for A, B and C be propositional variables?

Exercice 7 (Crolard's catch/throw calculus [Cro99]) *Show that classical logic can be axiomatised using the rules in place of the Focus and Unfocus rules.*

$$\frac{\Gamma \vdash_C \Delta_1, \alpha : A, \Delta_2; p : A}{\Gamma \vdash_C \Delta_1, A, \Delta_2; \mathsf{throw}_\alpha p : B}\ \mathit{Catch} \qquad \frac{\Gamma \vdash_C \Delta, A; p : A}{\Gamma \vdash_C \Delta; \mathsf{catch}_\alpha p : A}\ \mathit{Abort}$$

In particular, show how to derive a proof $\mu\alpha.[\beta]p$ using catch *and* throw*, and conversely, how to define* catch *and* throw *from μ and $[\,]$.*

Show that through the encoding, the following rules are derivable from the rules for μ:

$$\begin{array}{lcll} \mathsf{catch}_\alpha \mathsf{throw}_\alpha p & \to & \mathsf{catch}_\alpha p & \\ \mathsf{throw}_\alpha \mathsf{catch}_\beta & \to & \mathsf{throw}_\beta p[\alpha/\beta] & \\ \mathsf{throw}_\alpha \mathsf{throw}_\beta & \to & \mathsf{throw}_\beta p & \\ \mathsf{catch}_\alpha p & \to & p & \text{if } \alpha \text{ not free in } p \end{array}$$

Exercice 8 *Gentzen's original presentation of natural deduction is "context-free". Give a similar "context-free" formulation of Crolard's* catch/throw *calculus. Hint: one would need a way to represent the formulae of Δ as leaves in $\Gamma \vdash \Delta; A$, and one could use e.g. the notation $\overline{A}$ to mean the assumption of the refutation of A.*

Exercice 9 *Give similarly a "context-free" formulation of Parigot's classical natural deduction. Hint: one will need a way to represent sequents of the form $A_1, ...A_n\Gamma \vdash B_1, ..., B_p$; and one might consider that they prove a formal derivation of absurdity written $\perp\!\!\!\perp$, as in*

$$\begin{array}{ccccccc} A_1 & ... & A_n & \overline{B_1} & ... & \overline{B_p} & \\ & & & \vdots & & & \\ & & & \vdots & & & \\ & & & \perp\!\!\!\perp & & & \end{array}$$

Exercice 10 *Show that if we are only interested in a translation to intuitionistic logic, the translation $(A \Rightarrow B)^+$ can be shortened into $A^+ \Rightarrow B^+$ in Theorem 19.*

5.5 Historical remarks

The first translation was by Kolmogorov [Kol67] at the same time he designed the first logic for minimal logic. It was later rediscovered by Gödel [Göd32] to minimal logic and Gentzen [Gen36] who did not know about Kolmogorov's translation. Kuroda [Kur51] designed another translation to intuitionistic (non minimal) logic.

The connection between double-negation translation and continuation-passing-style translation was made by Murthy [Mur90] after knowing about Griffin's connection [Gri90] between classical logic and control operators.

After Griffin's result, Girard [Gir91] designed a global translation which highlights the role of polarities. In the context of the proof-as-program correspondence, Lafont, Reus and Streicher [LRS93] designed a translation working by dualising connectives which can be seen as an optimisation of Kolmogorov's translation: it simulates call-by-name and preserves η-equality. Krivine [Kri90] introduced a translation for the negative fragment of classical logic. In this case, only atoms have to be translated and Krivine replace them by their negation.

Analyses of the different double-negation translations were made by Ferreira and Oliva and Gaspar, investigating their relations and whether they target intuitionistic or minimal logic.

Analyses of the double-negation translation in terms of polarities has been conducted e.g. by Munch.

Chapter 6

Linear logic

6.1 Introduction to linear logic

Designed by Girard [Gir87], *Linear Logic*, shortly LL, is a formulation of logic highlighting a decomposition of the standard logical connectives into purely linear logical connectives and new explicit connectives for managing weakening and contraction. For instance, in linear logic, intuitionistic implication $A \Rightarrow B$ is decomposed into $!A \multimap B$ where $\multimap$ is a linear implication and $!A$ tells that A can be used as many times as one wants in proving B from A.

Besides the invention of the *resource* modality "!" and of its dual "?", an outcome of the work on linear logic is the distinction between two kinds of conjunctions, *tensor*, written $\otimes$, and *with*, written $\&$, as well as two kinds of disjunctions, *par*, written $\parr$, and *plus*, written $\oplus$.

The *tensor* connective $A \otimes B$ is a *multiplicative* conjunction: it combines the resources of a proof of A and of a proof of B. This is expressed by the right introduction rule

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma' \vdash \Delta', B}{\Gamma, \Gamma' \vdash \Delta, \Delta', A \otimes B}$$

where the contexts Γ and Γ', as well as Δ and Δ', seen as resources for the proof of A and B, are both kept. This is also expressed by the left introduction rule

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

which tells that using a resource $A \otimes B$ is the same as using both A and B. In particular, the left introduction rule tells that in a sequent $A_1, \ldots, A_n \vdash B_1, \ldots, B_p$, the implicit conjunctive meaning of the comma on the left is a tensor.

The *with* connective $A \,\&\, B$ is an *additive* conjunction: it asserts the possibility of a proof of A and of a proof of B but not of both of them at the same time. This is expressed by the right introduction rule

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \,\&\, B}$$

where only one copy of the context Γ and Δ, seen as resources for either the proof of A or a proof of B, is kept, leaving the ability to only have at the same time either a proof of A or a proof of B. This is also expressed by the two left introduction rules

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta} \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}$$

which say that using a proof of $A \,\&\, B$ is using either a proof of A or a proof of B but not both at the same time.

Dually, $\oplus$ is the *additive* disjunction and it has the ordinary intuitive meaning of disjunction, namely that a proof of $A \oplus B$ comes either from proof of A or from a proof of B as expressed by the following two right introduction rules which are dual to the left introduction rules of additive conjunction

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \oplus B,} \qquad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \oplus B}$$

Similarly, the left introduction rule for $\oplus$ expresses a case analysis, i.e. proving from $A \oplus B$ is equivalent to being able to prove from A and to prove from B

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}$$

While $\oplus$ and & are dual, the tensor connective is also involved in a duality. The dual of the tensor connective is the *par* connective. The par is an unusual connective in itself though it is sometimes used in proof search in order to have the right introduction rule of disjunction reversible. Indeed, the right introduction rule of par, dual to the left introduction rule of the tensor, is

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

Dually, the left introduction rule for par is

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \parr B \vdash \Delta, \Delta'}$$

which expresses that using a resource $A \parr B$ requires to split the existing side resources Γ, Γ' and Δ, Δ' into two sets of resources, one supporting the use of A and the other supporting the use of B.

Though unusual as itself, the par is however a key ingredient of linear implication, $\multimap$, which can be defined as $A \multimap B \triangleq A^\perp \parr B$ for $A^\perp$ an operation which replaces all connectives in A by their dual.

The resource modality !, called *bang*, or *of course*, is an *exponential* connective. A formula $!A$ is a formula which can be used as many times as wanted, what is expressed by the left rules for weakening and contraction

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}$$

The resource modality ! has a dual which is ?, called *why not*. A formula $?A$ is a formula whose proof is not necessarily linear, in the sense that A need not to be proved in exactly one "attempt", but instead in 0 or more than 2 "attempts", what is allowed by the right rules for weakening and contraction

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, ?A} \qquad \frac{\Gamma \vdash \Delta, ?A, ?A}{\Gamma \vdash \Delta, ?A}$$

The remaining left and right introduction rules for ! and ? are

$$\frac{!\Gamma \vdash ?\Delta, A}{!\Gamma \vdash ?\Delta, !A} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, ?A} \qquad \frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta}$$

Note that the right introduction rule for ! and the left introduction rule for ? require weakening and contraction to be freely allowed in the left and right contexts, what is expressed by expecting these contexts to be of the form $!\Gamma$ and $?\Delta$ respectively. This makes of ! and ? modal operators.

As an example of ? allowing to prove a formula in several attempts, let us consider the case of $A \oplus A^\perp$. The formular $A \oplus A^\perp$ is not provable in linear logic but $?(A \oplus A^\perp)$ is, and the proof proceeds by keeping a copy of $A \oplus A^\perp$, what the ? allows, then to assert $A^\perp$, which is the same as assuming A, then attempting another proof of $?(A \oplus A^\perp)$ by this time proving A, using the assumption A previously obtained (this is the rephrasing of the standard proof of excluded-middle in multi-conclusion sequent calculus, see page 30).

Finally, linear logic has four constants, 1, called *one*, 0, called *zero*, $\top$, called *true* and $\bot$, called *false*. They are the neutral elements for respectively $\otimes$, $\oplus$, & and $\parr$.

6.2 Duality in linear logic

The connectives of linear logic thus come by pairs, with *tensor* being the dual of *par*, *with* the dual of *plus*, and *of course* the dual of *why not*. For A a formula, the dual $A^\perp$ of A is an operation defined by the following clauses:

$$\begin{array}{lcl}
(A \,\&\, B)^\perp & \triangleq & A^\perp \oplus B^\perp \\
(A \otimes B)^\perp & \triangleq & A^\perp \parr B^\perp \\
(A \oplus B)^\perp & \triangleq & A^\perp \,\&\, B^\perp \\
(A \parr B)^\perp & \triangleq & A^\perp \otimes B^\perp \\
(!A)^\perp & \triangleq & ?A^\perp \\
(?A)^\perp & \triangleq & !A^\perp \\
0^\perp & \triangleq & \top \\
1^\perp & \triangleq & \bot \\
\top^\perp & \triangleq & 0 \\
\bot^\perp & \triangleq & 1
\end{array}$$

In particular, we have that $A^{\perp\perp}$ is identical to A.

6.3 A classification of linear logic connectives

All in all the connectives of linear logic can be classified according to different criteria which are summarized in the following table:

	additive	multiplicative	exponential
positive	$\oplus/0$	$\otimes/1$	!
negative	$\&/\top$	$⅋/\bot$	?

In particular, the *positive* connectives are those whose left introduction rule is reversible (i.e. that the premises can be derived from the conclusion) while the *negative* connectives are those for which it is the right introduction rule which is reversible.

6.4 Inference rules

Linear Logic decomposes standard connectives into primitive linear connectives whose semantics, as expressed by the inference rules, is precise. As said in the introduction, intuitionistic implication $A \Rightarrow B$ is decomposed into $!A \multimap B$, and $A \multimap B$ can itself be rephrased in terms of the *par* connective as $A \multimap B \triangleq A^{\perp} ⅋ B$, leading to the following derived inference rules for $\multimap$:

$$\frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \multimap B} \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \multimap B \vdash \Delta, \Delta'}$$

By combining all possibilities of taking a formula on one side of a sequent and the other also on one side of a sequent, one thus gets four different variants of $A ⅋ B$, namely $A ⅋ B$ itself when both formulas come from the same side, $A \multimap B$ as above when A comes from the left-hand side, a connective that we could write $A \circ\!\!- B$ when it is B which comes from the left-hand side, and a connective which we could write $A \mathbin{⅋^{\circ}} B$ as it is a linear version of Sheffer stroke $A \uparrow B$ (i.e. a not and). Similarly, one can consider three variants of each of $A \otimes B$, $A \oplus B$ and $A \,\&\, B$, which, all in all, would give twelve connectives, thus twelve left introduction rules and twelve right introduction rules.

To avoid this proliferation, it is customary to fold linear logic onto a calculus with one-sided sequents of the form $\vdash \Gamma$, joining at the same time the introduction rules for $\&$ and $\oplus$ as well as the introduction rules for $⅋$ and $\otimes$, as well as the introduction rules for ? and !, drastically reducing the number of rules.

Let us also consider quantifiers which can be seen as dependent additive connectives, therefore written $\oplus_x A$ and $\&_x A$. The syntax of formulae is thus:

$$A, B, C \quad ::= \quad X(t_1, ..., t_{\mathsf{ar}_X}) \mid 0 \mid 1 \mid \top \mid \bot \mid A \otimes A \mid A ⅋ A \mid A \oplus A \mid \& \mid \oplus_x A \mid \&_x A$$

The syntax of contexts is as usual:

$$\Gamma \quad ::= \quad \emptyset \mid \Gamma, A$$

The resulting logic is presented in Figure 6.1 where contexts are considered up to permutations. Formulas marked with ? are erasable/duplicable. The connectives ! and ? behave like modalities, in that the promotion rule expects that all other formulae are marked as erasable/duplicable.

Like in Gentzen's sequent calculus, proofs of linear logic come with a rewriting system to eliminate cuts.

There is also a procedure of η-expansion for axioms, so that axioms apply to atoms only. For instance, turned in two-sided sequent calculus, the axiom

$$\frac{}{X \otimes Y \vdash X \otimes Y}\, Ax$$

expands to

$$\dfrac{\dfrac{\dfrac{}{X \vdash X}\, Ax \qquad \dfrac{}{Y \vdash Y}\, Ax}{X, Y \vdash X \otimes Y}}{X \otimes Y \vdash X \otimes Y}$$

6.5 Properties of connectives

In addition to the usual associativity and commutativity of conjunction and disjunction, linear logic satisfies various logical isomorphisms (written $A \circ\!\!-\!\!\circ B$ and meaning $A \vdash B$ and $B \vdash A$), some of them being even proof isomorphisms (written $A \simeq B$ and meaning not only $A \vdash B$ and $B \vdash A$ but that a cut between the corresponding proofs reduces to an axiom rule, up to η-expansion):

rules about the structure

$$\frac{}{\vdash A, A^{\perp}}\ Ax$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma', A^{\perp}}{\vdash \Gamma, \Gamma'}\ Cut$$

rules defining multiplicative connectives

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma', B}{\vdash \Gamma, \Gamma', A \otimes B}\ \otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ \parr \qquad \frac{}{\vdash 1}\ 1 \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\ \perp$$

rules defining additive connectives and quantifiers

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\ \oplus_1 \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \oplus_2 \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\ \& \qquad \text{(no rule for 0)} \qquad \frac{}{\vdash \Gamma, \top}\ \top$$

$$\frac{\vdash \Gamma, A[t/x]}{\vdash \Gamma, \oplus_x A}\ \oplus_x \qquad \frac{\vdash \Gamma, A \qquad x \notin FV(\Gamma)}{\vdash \Gamma, \&_x A}\ \&_x$$

rules for exponential modalities, supporting weakening and contraction

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ promotion \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ dereliction \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ weakening \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ contraction$$

Figure 6.1: Linear Logic, one-sided

distributivity

$$\begin{array}{lcl} A \otimes (B \oplus C) & \simeq & (A \otimes B) \oplus (A \otimes C) \\ (B \oplus C) \otimes A & \simeq & (B \otimes A) \oplus (C \otimes A) \\ A \parr (B \,\&\, C) & \simeq & (A \parr B) \,\&\, (A \parr C) \\ (B \,\&\, C) \parr A & \simeq & (B \parr A) \,\&\, (C \parr A) \end{array}$$

associativity

$$\begin{array}{lcl} A \otimes (B \otimes C) & \simeq & (A \otimes B) \otimes C \\ A \parr (B \parr C) & \simeq & (A \parr B) \parr C \\ A \oplus (B \oplus C) & \simeq & (A \oplus B) \oplus C \\ A \,\&\, (B \,\&\, C) & \simeq & (A \,\&\, B) \,\&\, C \end{array}$$

commutativity

$$\begin{array}{lcl} A \otimes B & \simeq & B \otimes A \\ A \parr B & \simeq & B \parr A \\ A \oplus B & \simeq & B \oplus A \\ A \,\&\, B & \simeq & B \,\&\, A \end{array}$$

neutral elements

$$\begin{array}{lcl} A \otimes 1 & \simeq & A \\ 1 \otimes A & \simeq & A \\ A \oplus 0 & \simeq & A \\ 0 \oplus A & \simeq & A \\ A \parr \bot & \simeq & A \\ \bot \parr A & \simeq & A \\ A \,\&\, \top & \simeq & A \\ \top \,\&\, A & \simeq & A \end{array}$$

exponential structure

$$\begin{array}{lcl} ?(A \oplus B) & \simeq & ?A \parr ?B \\ !(A \,\&\, B) & \simeq & !A \otimes !B \end{array}$$

idempotencies

$$\begin{array}{lcl} ?(?A \& ?B) & \multimap & ?A \& ?B \\ ?(?A \parr ?B) & \multimap & ?A \parr ?B \\ ??A & \multimap & ?A \\ !(!A \oplus !B) & \multimap & !A \oplus !B \\ !(!A \otimes !B) & \multimap & !A \otimes !B \\ !!A & \multimap & !A \end{array}$$

More precisely, the idempotencies forms retraction/section, in the sense that only one of the two cuts between the two proofs of the logical isomorphisms reduces to an axiom rule. For instance, the composition of the proof of $??A \vdash ?A$ with each of the two proofs of $?A \vdash ??A$ leads to a proof of $??A \vdash ??A$ which first applies dereliction on the right (starting from the root) while the η-expansion of the axiom first applies promotion on the left.

We also have the following properties which hold in only one direction:

contraction and weakening

$!A$	$\multimap$	1	weakening
$!A$	$\multimap$	A	derelection
$!A$	$\multimap$	$!A \otimes !A$	contraction
$\bot$	$\multimap$	$?A$	weakening
A	$\multimap$	$?A$	derelection
$?A \parr ?A$	$\multimap$	$?A$	contraction

weak distributivity of exponential

$!A \otimes !B$	$\multimap$	$!(A \otimes B)$	comonad ! is monoidal for $\otimes$ (1)
1	$\multimap$	$!1$	comonad ! is monoidal for $\otimes$ (2)
$?(A \parr B)$	$\multimap$	$?A \parr ?B$	monad ? is comonoidal for $\parr$ (1)
$?\bot$	$\multimap$	$\bot$	monad ? is comonoidal for $\parr$ (2)
$!A \oplus !B$	$\multimap$	$!(A \oplus B)$	
$?(A \mathbin{\&} B)$	$\multimap$	$?A \mathbin{\&} ?B$	

weak distributivity of binary connectives

$(A \mathbin{\&} B) \otimes C$	$\multimap$	$(A \otimes C) \mathbin{\&} (B \otimes C)$
$(A \mathbin{\&} B) \oplus C$	$\multimap$	$(A \oplus C) \mathbin{\&} (B \oplus C)$
$(A \parr C) \oplus (B \parr C)$	$\multimap$	$(A \oplus B) \parr C$
$(A \mathbin{\&} C) \oplus (B \mathbin{\&} C)$	$\multimap$	$(A \oplus B) \mathbin{\&} C$
$(A \parr B) \otimes (C \otimes D)$	$\multimap$	$(A \otimes C) \parr (B \otimes D)$
$(A \parr C) \otimes (B \parr D)$	$\multimap$	$(A \otimes B) \parr (C \parr D)$

weak associativity[1]

$A \otimes (B \parr C)$	$\multimap$	$(A \otimes B) \parr C$

6.6 Synthetic connectives

Chapter 7

The computational core of proofs and programs: polarisation

It is common in sequent calculus based proof search to decompose connectives into connectives whose introduction rules are *reversible* and connectives whose introduction rules are *irreversible* [**?**].

This influenced the approach to linear logic via Andreoli's design of a Prolog based on linear logic [And92]. Andreoli introduced the terminology *asynchronous* for a connective whose left introduction rule is reversible in sequent calculus and *synchronous* for a connective whose right introduction rule is reversible. Girard called them respectively of *negative* and *positive* polarity. This notion of polarity was used in his study of double-negation translations for classical logic [Gir91]. The decomposition of formulae into positive and negative phases was crystallised by Laurent [Lau02] into a proper variant of linear logic dubbed *Polarised Linear Logic* (LLP) which can in turned be presented as the fragment LJ_0 of Gentzen's LJ made from positive connectives and negation. LLP was further decomposed by Melliès and Tabareau [MT10] under the name *Tensorial Logic* (TL) in order to isolate the purely polarised part from the weakening/contraction structure.

Polarised logic can be presented using a duality between positive and negative connectives, as in the case of (one-sided) LLP or one-sided tensorial logic, or with an asymmetric two-sided bias highlighting either positive formulae (LJ_0) or negative formulae (the dual LD_0 to LJ_0), or with a symmetric two-sided logic considering both positive and negative connectives living in independent world

Polarised logic can also be presented as a sequent calculus or as a natural deduction. For instance, the natural deduction counterpart NJ_0 to LJ_0 was proposed by Curien.

Proofs of polarised logic can be more or less constrained. In the most constrained form, called focused, as e.g. in Liang and Miller's LJF and LKF [LM09] or Zeilberger's focused calculus [Zei08] the negative phases have a canonical structure, while in the less constrained forms, e.g. LLP or TL, the positive phase is constrained but the negative rules can happen at any time, even intertwined with a positive phase.

The positive and negative connectives can also be considered synthetically as in ludics [Gir01] (see also [**?**]) or made from elementary bricks (the units, the binary positive disjunction and conjunction, the existential quantifier, as well as negation or implementation in the two-sided cases).

The transition from the positive to the negative phase can be explicit as in LLP, TL, Zeilberger's calculus, ludics, or implicit as in LJF, LKF, or in Munch-Maccagnoni's $\mathrm{L}_{\mathsf{pol}}$ variant [**?**] of a polarised variant of LK called LK^{η}_{p} from Danos, Joinet and Schellinx [DJS95].

Inspired by a categorical approach to program semantics, Lévy independently developed a polarised typed λ-calculus dubbed *call-by-push-value* (CBPV) [Lev99]. Retrospectively, CBPV can be seen as a fragment of a symmetric two-sided polarised logic in natural deduction form, with no strong focusing constraints on the negative phase, and the connective made from standard bricks.

The treatment of contraction and weakening can be explicit and an autonomous as in TL, or in Munch-Maccagnoni's polarised presentation of Multiplicative Additive Linear Logic. It can be explicit and attached to the change of phase as in LLP. It can be implicit and, when implicit, in an intuitionistic style, as in LJF and CBPV, or in a classical style, as in LKF and $\mathrm{L}_{\mathsf{pol}}$.

Besides call-by-push-value and Munch-Maccagnoni's $\mathrm{L}_{\mathsf{pol}}$, none of the calculus cited above come with a proof-as-program presentation.

We shall present in this chapter various "minimalistic" such calculi whose operational semantics is canonical, in the sense that cuts have a canonical operational semantics based on the form of the constructors.

We shortly summarise the differences and equivalences between systems.

7.1 The core of computation: the continuation-passing-style target (linear part)

We present in this section a minimalist calculus in which any kind of computational behaviour can be simulated up to an encoding. We shall call it *computational core logic* . It is the target of continuation-passing-style translation, with sequents of the form $\Gamma \vdash$, standing implicitly for $\Gamma \vdash \bot$, as well as $\Gamma \vdash P$ for P restricted to a set of positive formulae, as well as, finally $\Gamma; P \vdash$ indicating a focus on P. This logic is also a subset of the asymmetric two-sided proof-as-program presentation of the weakening/contraction-free core of both Laurent's polarised linear logic and of Melliès and Tabareau's tensorial logic (up to the additional sequents with a focus).

Formulae are only positive and built from the following grammar:

$$P, Q, R \quad ::= \quad X(t_1, ..., t_{\mathsf{ar}_X}) \mid 0 \mid 1 \mid P \oplus Q \mid P \otimes Q \mid \oplus_x P \mid \neg P$$

where we used names inherited from linear logic to emphasise that the connectives come equipped with the operational and observational meaning semantics of linear logic.

Typing contexts are built from positive formulae:

$$\Gamma \quad ::= \quad \emptyset \mid \Gamma, a : P$$

The calculus is based on three kinds of derivations: values, evaluation contexts and commands. The syntax is:

$$\begin{array}{lll} V & ::= & a \mid () \mid (V, V') \mid \iota_1(V) \mid \iota_2(V) \mid (t, V) \mid \neg e \\ e & ::= & \tilde{\mu}a.c \mid \tilde{\mu}().c \mid \tilde{\mu}() \mid \tilde{\mu}(a, b).c \mid [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] \mid \tilde{\mu}(x, a).c \mid \neg V \\ c & ::= & \langle V \parallel e \rangle \end{array}$$

The calculus is based on three kinds of sequents:

- $\Gamma \vdash V : P$ for proofs-as-programs
- $\Gamma; e : P \vdash$ for evaluation contexts
- $\Gamma \vdash c$ for proofs-as-commands

The inference rules, annotated by terms, are given on Figure 7.1.
Finally, reduction rules are:

computational rules

$$\begin{array}{lll} (\beta_1) & \langle () \parallel \tilde{\mu}().c \rangle & \stackrel{h}{\to} \quad c \\ (\beta_\otimes) & \langle (V_1, V_2) \parallel \tilde{\mu}(a_1, a_2).c \rangle & \stackrel{h}{\to} \quad c[V_1/a_1][V_2/a_2] \\ (\beta^1_\oplus) & \langle \iota_1(V) \parallel [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] \rangle & \stackrel{h}{\to} \quad c_1[V/a] \\ (\beta^2_\oplus) & \langle \iota_2(V) \parallel [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] \rangle & \stackrel{h}{\to} \quad c_2[V/a] \\ (\beta_{\oplus_x}) & \langle (t, V) \parallel \tilde{\mu}(x, a).c \rangle & \stackrel{h}{\to} \quad c[t/x][V/a] \\ (\beta_\neg) & \langle \neg e \parallel \neg V \rangle & \stackrel{h}{\to} \quad \langle V \parallel e \rangle \\ (\beta_{\tilde{\mu}}) & \langle V \parallel \tilde{\mu}a.c \rangle & \stackrel{h}{\to} \quad c[V/a] \end{array}$$

observational rules

$$\begin{array}{lllll} (\eta_1) & \tilde{\mu}().\langle () \parallel e \rangle & = & e & \\ (\eta_\otimes) & \tilde{\mu}(a_1, a_2).\langle (a_1, a_2) \parallel e \rangle & = & e & a \notin FV(e) \\ (\eta_\oplus) & [\tilde{\mu}a.\langle \iota_1(a) \parallel e \rangle | \tilde{\mu}a.\langle \iota_2(a) \parallel e \rangle] & = & e & a \notin FV(e) \\ (\eta_{\oplus_x}) & \tilde{\mu}(x, a).\langle (x, a) \parallel e \rangle & = & e & a \notin FV(e), x \notin FV_t(e) \\ (\eta_\neg) & \neg\neg e & = & e & \\ (\eta_{\tilde{\mu}}) & \tilde{\mu}a.\langle a \parallel e \rangle & = & e & a \notin FV(e) \end{array}$$

7.2 Properties

Note that the set of connectives is complete for classical predicate logic, as one can interpret the linear connectives as follows:

structural rules

$$\frac{}{a : P \vdash a : P}$$

$$\frac{\Gamma, a : P \vdash c}{\Gamma; \tilde{\mu}a.c : P \vdash}$$

$$\frac{\Gamma \vdash V : P \qquad \Gamma'; e : P \vdash}{\Gamma, \Gamma' \vdash \langle V \parallel e \rangle}$$

rules defining multiplicative connectives

$$\frac{}{\vdash () : 1} \qquad \frac{\Gamma \vdash V : P \qquad \Gamma' \vdash V' : Q}{\Gamma, \Gamma' \vdash (V, V') : P \otimes Q}$$

$$\frac{}{\Gamma; \tilde{\mu}() : 0 \vdash} \qquad \frac{\Gamma \vdash c}{\Gamma; \tilde{\mu}().c : 1 \vdash} \qquad \frac{\Gamma, a : P, b : Q \vdash c}{\Gamma; \tilde{\mu}(a, b).c : P \otimes Q \vdash}$$

rules defining additive connectives and quantifiers

$$\frac{\Gamma \vdash V : P}{\Gamma \vdash \iota_1(V) : P \oplus Q} \qquad \frac{\Gamma \vdash V : Q}{\Gamma \vdash \iota_2(V) : P \oplus Q} \qquad \frac{\Gamma \vdash V : P[t/x]}{\Gamma \vdash (t, V) : \oplus_x P}$$

$$\frac{\Gamma, a : P \vdash c_1 \qquad \Gamma; a : Q \vdash c_2}{\Gamma; [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] : P \oplus Q \vdash} \qquad \frac{\Gamma, a : P[t/x] \vdash c}{\Gamma; \tilde{\mu}(x, a).c : \oplus_x P \vdash}$$

rules for explicit change of polarity

$$\frac{\Gamma; e : P \vdash}{\Gamma \vdash \neg e : \neg P} \qquad \frac{\Gamma \vdash V : P}{\Gamma; \neg V : \neg P \vdash}$$

Figure 7.1: Core computational logic

$$
\begin{array}{lcl}
(A \wedge B)^* & \triangleq & A^* \otimes B^* \\
(A \vee B)^* & \triangleq & A^* \oplus B^* \\
(A \Rightarrow B)^* & \triangleq & \neg(A^* \otimes \neg B^*) \\
(\exists x\, A)^* & \triangleq & \oplus_x A^* \\
(\forall x\, A)^* & \triangleq & \neg \oplus_x \neg A^*
\end{array}
$$

Even if we do not consider the calculus interesting from a logical point of view as the hypothesis/conclusion status of usual connectives collapses through de Morgan laws, the calculus can express the standard connectives predicate calculus up to the embedding

Theorem 20 (Logical strength) *If $\Gamma \vdash A$ is provable then $\Gamma^*, \neg(A^*)^\perp \vdash$.*

Confluence for the operational rules comes easily since the system has no critical pairs.

Theorem 21 (Confluence) *If $p \twoheadrightarrow q$ and $p \twoheadrightarrow q'$ then there exists p' such that $q \twoheadrightarrow p'$ and $q' \twoheadrightarrow p'$.*

Termination of the operational rules come easily

Theorem 22 (Termination) *If $\Gamma \vdash p : P$ (resp. $\Gamma; e : P \vdash$ or $\Gamma \vdash c$) then p (resp. e, c) normalises.*

Remark 11 *Alternative design choices can be made in the definition of the calculus. We mention a few of them.*

- *The evaluation context for $\oplus$ could have been taken to be $[e|e']$ together with the following introduction rule:*

$$\frac{\Gamma; e : P \vdash \qquad \Gamma; e' : Q \vdash}{\Gamma; [e|e'] : P \oplus Q \vdash}$$

  Both presentations are mutually derivable the one from the other, up to a use of $\eta_{\tilde{\mu}}$. We however keep the longer form which is more consistent with the other left introduction rules.

- *There are alternative choices in notations. For instance, here is another system of notations which highlights the programming view:*

$$
\begin{array}{lll}
\textbf{let } () = [\,] \textbf{ in } c & \textit{for} & \tilde{\mu}().c \\
\mathsf{efq}\ [\,] & \textit{for} & \tilde{\mu}() \\
\mathsf{dest}\ [\,] \ \mathsf{as}\ (a,b)\ \mathsf{in}\ c & \textit{for} & \tilde{\mu}(a,b).c \\
\mathsf{case}\ [\,] \ \mathsf{of}\ [\iota_1(a) \to c | \iota_2(a') \to c'] & \textit{for} & [\tilde{\mu}a.c | \tilde{\mu}a'.c] \\
\mathsf{dest}\ [\,] \ \mathsf{as}\ (x,a)\ \mathsf{in}\ c & \textit{for} & \tilde{\mu}(x,a).c \\
\textbf{let } a = [\,] \textbf{ in } c & \textit{for} & \tilde{\mu}a.c \\
\neg V[\,] & \textit{for} & \neg V
\end{array}
$$

  and $\langle V \parallel e \rangle$ being represented as V filling the hole $[\,]$ of e, i.e. as $e[V]$.

  With these latter notations, the reduction rules show as follows:

$$
\begin{array}{lll}
(\beta_1) & \textbf{let } () = () \textbf{ in } c & \xrightarrow{h} \ c \\
(\beta_\otimes) & \mathsf{dest}\ (V_1,V_2)\ \mathsf{as}\ (a_1,a_2)\ \mathsf{in}\ c & \xrightarrow{h} \ c[V_1/a_1][V_2/a_2] \\
(\beta^1_\oplus) & \mathsf{case}\ \iota_1(V)\ \mathsf{of}\ [\iota_1(a_1) \to c_1 | \iota_2(a_2) \to c_2] & \xrightarrow{h} \ c_1[V/a_1] \\
(\beta^2_\oplus) & \mathsf{case}\ \iota_2(V)\ \mathsf{of}\ [\iota_1(a_1) \to c_1 | \iota_2(a_2) \to c_2] & \xrightarrow{h} \ c_2[V/a_2] \\
(\beta_\exists) & \mathsf{dest}\ (t,V)\ \mathsf{as}\ (x,a)\ \mathsf{in}\ c & \xrightarrow{h} \ c[t/x][V/a] \\
(\beta_\neg) & \neg V \neg e & \xrightarrow{h} \ e[V] \\
(\beta_\mu) & \textbf{let } a = V \textbf{ in } c & \xrightarrow{h} \ c[V/a]
\end{array}
$$

- *There are alternative choices in reduction rules. For instance, the substitution could be done only at the time of $\langle V \parallel \tilde{\mu}a.c \rangle$, as follows*

$$
\begin{array}{lll}
(\beta_1) & \langle () \parallel \tilde{\mu}().c \rangle & \xrightarrow{h} \ c \\
(\beta_\otimes) & \langle (V_1,V_2) \parallel \tilde{\mu}(a_1,a_2).c \rangle & \xrightarrow{h} \ \langle V_1 \parallel \tilde{\mu}a_1.\langle V_2 \parallel \tilde{\mu}a_2.c \rangle \rangle \\
(\beta^1_\oplus) & \langle \iota_1(V) \parallel [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] \rangle & \xrightarrow{h} \ \langle V \parallel \tilde{\mu}a.c_1 \rangle \\
(\beta^1_\oplus) & \langle \iota_2(V) \parallel [e_1|e_2] \rangle & \xrightarrow{h} \ \langle V \parallel \tilde{\mu}a.c_2 \rangle \\
(\beta_\exists) & \langle (t,V) \parallel \tilde{\mu}(x,a).c \rangle & \xrightarrow{h} \ \langle V[t/x] \parallel \tilde{\mu}a.c \rangle \\
(\beta_\neg) & \langle \neg e \parallel \neg V \rangle & \xrightarrow{h} \ \langle V \parallel e \rangle \\
(\beta_\mu) & \langle V \parallel \tilde{\mu}a.c \rangle & \xrightarrow{h} \ c[V/a]
\end{array}
$$

- *There is an alternative choice in dealing with the cut rule. With the chosen design, cuts of the form* $\langle a \parallel e \rangle$*, when* e *is not a* $\tilde{\mu}b.c$*, are not eliminable.*

  The alternative is to extend commands:

$$c \quad ::= \ldots \mid a\, e$$

  together with the inference rule:

$$\frac{\Gamma; e : P \vdash}{\Gamma, a : P \vdash a\, e}$$

  and the reduction rule:

$$(\beta_{var}) \quad \langle a \parallel e \rangle \quad \xrightarrow{h} \quad a\, e \quad e \text{ not of the form } \tilde{\mu}a.c$$

  The price to pay is however a more complex definition of substitution, since substitution of variables would occur not only at axioms but also at the new rule. This justifies the preference to keeping a cut which is not eliminable.

7.3 The core of computation: the continuation-passing-style target (non-linear extension)

7.4 LJ_0, DL_0 and LLP

The calculi LJ_0, DL_0 and LLP are three isomorphic presentations of the same calculus differing in whether formulae are placed in position of hypotheses or of conclusions.

The calculus LJ_0 is the restriction of Gentzen's LJ to positive connectives, which we shall write 0, 1, $\oplus$, $\otimes$ and $\oplus_x$ in reference to linear logic. It extends the core computational logic with sequents of the form $\Gamma \vdash P;$ and $\Gamma; Q \vdash P;$ supporting unfocusing.

The syntax is as in the core computational logic but extended with two constructions for

$$\begin{array}{lcl} V & ::= & a \mid () \mid (V, V') \mid \iota_1(V) \mid \iota_2(V) \mid (t, V) \mid \neg e \\ e & ::= & \tilde{\mu}a.c \mid \tilde{\mu}().c \mid \tilde{\mu}() \mid \tilde{\mu}(a,b).c \mid [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] \mid \tilde{\mu}(x,a).c \mid \neg V \\ t & ::= & \tilde{\mu}().t \mid \tilde{\mu}(a,b).t \mid [\tilde{\mu}a.t_1 | \tilde{\mu}a.t_2] \mid \tilde{\mu}(x,a).t \\ c & ::= & \langle V \parallel e \rangle \mid \langle t \parallel \star \rangle \end{array}$$

The inference rules are as in Figure 7.1 but with extra rules

$$\frac{\Gamma \vdash t : P}{\Gamma \vdash P; \langle t \| \star \rangle} \qquad \frac{\Gamma \vdash P; c}{\Gamma \vdash \mu \star .c : P}$$

$$\frac{\Gamma \vdash c}{\Gamma; \tilde{\mu}().c : 1 \vdash} \qquad \frac{\Gamma, a : P, b : Q \vdash c}{\Gamma; \tilde{\mu}(a,b).c : P \otimes Q \vdash} \qquad \frac{\Gamma, a : P \vdash c_1 \qquad \Gamma; a : Q \vdash c_2}{\Gamma; [\tilde{\mu}a.c_1 | \tilde{\mu}a.c_2] : P \oplus Q \vdash} \qquad \frac{\Gamma, a : P[t/x] \vdash c}{\Gamma; \tilde{\mu}(x,a).c : \oplus_x P \vdash}$$

LJ_0 can be interpreted into core computational logic as follows:

$$\Gamma \vdash p : P$$

Chapter 8

Markov's principle

8.1 Conservativity of classical logic over intuitionistic logic for Σ_1^0-formulae

8.2 Markov's rule

8.3 Friedman's A-translation

We shall give several variants of Friedman's A-translation, starting with the original translation for historical purpose, and continuing with variants which reflect the algebraic structure of the translation. Note that Dragalin designed a similar translation on the other side of iron curtain [Dra80]. A preliminary translation also occurs in Prawitz and Malmnäs [PM68].

8.3.1 The original Friedman's A-translation

Let A be a fixed formula. The original A-translation by Friedman [Fri78] is the following transformation of formulae which replace any atomic formula by its disjunction with A and otherwise is compositional.

$$\begin{array}{lcl}
X(t_1,..,t_n)^A_{Fri} & \triangleq & X(t_1,...,t_n) \vee A \\
\bot^A_{Fri} & \triangleq & \bot \vee A \\
\top^A_{Fri} & \triangleq & \top \vee A \\
B \wedge C^A_{Fri} & \triangleq & \wedge^B_{Fri} C^A_{Fri} \\
B \vee C^A_{Fri} & \triangleq & \vee^B_{Fri} C^A_{Fri} \\
B \Rightarrow C^A_{Fri} & \triangleq & \Rightarrow^B_{Fri} C^A_{Fri} \\
\exists x\, B^A_{Fri} & \triangleq & \exists x\, B^A_{Fri} \\
\forall x\, B^A_{Fri} & \triangleq & \forall x\, B^A_{Fri}
\end{array}$$

and we write Γ^A_{Fri} for the piece-wise extension of the translation to contexts.

Of course, one could as well have translated $\bot$ to A which is classically equivalent to $\bot \vee A$ and $\top$ to $\top$ which is classically equivalent to $\top \vee A$, but we preserve the original translation for historical reason.

The main point of Friedman's A-translation is to obtain the following result:

Theorem 23 *$\Gamma \vdash_I B$ implies $\Gamma^A_{Fri} \vdash_M B^A_{Fri}$*

8.4 Coquand-Hofmann's A-translation

8.5 Conservativity of classical logic over intuitionistic logic for positive formulae

8.6 Markov's rule in propositional logic

8.7 Internalisation of Markov's rule

Chapter 9

The computational content of sequent calculus

Sequent calculus was designed in 1935 by Gentzen [Gen35] so as to be able to show that proofs have normal forms. Normal proofs are characterised by the absence of "cuts", which is the only rule of the calculus allowing "detours". Otherwise said, detours can be eliminated.

Even though the 20s and 30s saw the birth of various computational models, namely combinatory logic with Schönfinkel and Curry, λ-calculus with Church, recursive functions with Gödel, Turing's machines with Turing, even though combinatory logic and λ-calculus were first motivated by the foundations of logic (to represent formulae and in particular quantifiers) before being used as a foundation of programming after paradoxes were found in the initial versions by Schönfinkel and Church, even though Curry had noticed as early as 1934 that the basic combinators of combinatory logic were related to the axioms of Frege-Hilbert systems, there is no sign of awareness in the literature at this time, neither that natural deduction and λ-calculus could be related, nor that cut-elimination was after all the first form of computational interpretation available for a formal system.

In particular, it is only from the 90's that a couple of works contributed to interpret sequent calculus not only as a logic but also as a computational model. Beside works leading to indirectly interpret the rules of sequent calculus using λ-calculus, one of the objective was to give a direct computational interpretation to the left introduction rules. Kesner, Puel and Val Breazu-Tannen [KPT96] interpreted the left introduction rules as pattern-matching operators while [Her95a, Her95b] focused on the interpretation of implication and interpreted the left introduction rules of implication as an operator to build list of arguments. Only a subsystem of full sequent calculus of (Kleene's G3 variant of) Gentzen's LK was attained, and, later on, an interpretation covering the whole calculus was given [CH98]. We shall describe it below.

Remark 12 (About the terminology "sequent calculus") *The term "sequent calculus" is source of confusion. In the original work by Gentzen [Gen35], the calculi were called* Logistische Kalküle. *The terminology "sequent calculus" started to become popular in the 60's: Prawitz [Pra65] uses the terminology "calculus of sequents" by contrast with natural deduction which was not known at this time to be expressible as a calculus of "sequents", i.e., as a calculus manipulating judgements of the form $\Gamma \vdash A$; Troelstra [Tro73] uses the terminology "sequent calculus". On his side, Kleene [Kle62], who designed a variant of Gentzen's* Logistische Kalküle *he called G3, is using the term "Gentzen's systems" which he opposes to "Hilbert-type systems"*[1]

On the other side, Howard [How69] uses the term "sequent calculus" to mean natural deduction formulated with sequents, somehow indicating a confusion in the 60's on whether it was the structure of the proof system (right introduction and right elimination vs right introduction and left introduction) or the style of presentation (as a tree annotated by formulae or as a tree annotated by sequents) which was relevant in the system under consideration.

9.1 On the general layout of sequent calculus

There have been various ways to present sequent calculus in the literature since Gentzen, depending on how contexts of formulae are represented (as lists, as sets, as multisets), depending on the use of structural rules (whether weakening or contraction are implicit or not, whether exchange is needed or not), depending on the exact choice of rules for connectives (positive or negative conjunction and disjunctions), depending on how the two sides of a sequent are used (two-sided, mono-sided, ...).

In order to assign a computational content to sequent calculus, we shall give a presentation which uses lists (as we did for natural deduction), with weakening implicit in the axiom rules, with contraction implicit in the multi-premises

[1] Kleene calls G1 the original system from Gentzen and G2 the variant where the cut rule is replaced with a mix rule, as introduced by Gentzen in his proof of cut-elimination.

inference rules, with no exchange rule needed. For the connectives, we shall consider a positive (hence additive) disjunction but consider both a negative (additive) and a positive (multiplicative) conjunction. As long as symmetry is the concern, we shall use the left-hand side of sequents for hypotheses and the right-hand side of sequents for conclusions, but when it comes to adapt sequent calculus to dependent types, having all of hypotheses and conclusions on the left-hand side might be needed.

As a matter of comparison, Gentzen's original presentation was representing contexts with lists. He had explicit exchange, weakening and contraction rules (even though exchange rules are actually not needed per se). He had a positive disjunction (i.e. an additive one) and a negative conjunction (i.e. also an additive one).

As another example, Kleene's system G3 is using lists for contexts, has weakening attached to the axiom rule, contraction attached to each introduction rule, and no exchange rule. The disjunction is positive and the conjunction negative.

9.2 Towards decorating sequent calculus with a term notation

The first key ingredient which we shall be using to give a computational interpretation to sequent calculus is to possibly distinguish an active formula in the sequents. We shall then have three kinds of sequents:

- sequents with an active formula on the right written $\Gamma \vdash \Delta; A$;
- sequents with an active formula on the left written $\Gamma; A \vdash \Delta; A$;
- sequents with no active formula written $\Gamma \vdash \Delta$.

We shall then see sequents of the form $\Gamma \vdash \Delta; A$ as proofs of A under hypotheses in Γ and possibly proving side conclusions in Δ. We shall see sequents of the form $\Gamma; A \vdash \Delta$ as counterproofs of A, which we could also call co-proofs of A, also under hypotheses in Γ and possibly proving side conclusions in Δ. We shall finally see sequents of the form $\Gamma \vdash \Delta$ as interaction, or cut, between a proof and a counterproof.

We shall associate terms p to proofs, evaluation contexts e to counterproofs, and a syntactic category called commands and written c to cuts. Otherwise said, we shall have the following three kinds of annotated sequents

- $\Gamma \vdash p : A; \Delta$
- $\Gamma; e : A \vdash \Delta$
- $c : (\Gamma \vdash \Delta)$

9.3 The core annotated structure of sequent calculus

In a first step, we shall consider the core "computational" structure of sequent calculus, without considering any connective. Like classical natural deduction, this core has two kinds of variables: variables for terms, ranged over by a, b, etc. and variables for evaluation contexts ranged over by α, β, ...

$$\begin{array}{lrcl} \text{Commands} & c & ::= & \langle v \parallel e \rangle \\ \text{Terms} & p & ::= & a \mid \mu\alpha.c \\ \text{Evaluation contexts} & e & ::= & \alpha \mid \tilde{\mu}x.c \end{array}$$

The inference rules are the following:

$$Ax_L \ \frac{\alpha : A \in \Delta}{\Gamma; \alpha : A \vdash \Delta} \qquad Ax_R \ \frac{a : A \in \Gamma}{\Gamma \vdash a : A; \Delta}$$

$$\tilde{\mu} \ \frac{c : (\Gamma, a : A \vdash \Delta)}{\Gamma; \tilde{\mu}a.c : A \vdash \Delta} \qquad \mu \ \frac{c : (\Gamma \vdash \Delta, \alpha : A)}{\Gamma \vdash \mu\alpha.c : A; \Delta}$$

$$Cut \ \frac{\Gamma \vdash p : A; \Delta \qquad \Gamma; e : A \vdash \Delta}{\langle v \parallel e \rangle : (\Gamma \vdash \Delta)}$$

Note the presence of two axioms, one with an active formula on the right, annotated by a proof, that is, from a computational perspective, a program, and one with an active formula on the left, annotated by a counter-proof, that is, from a computational perspective, an evaluation context.

Note that a cut, represented by a command, defined an interaction between a program and an evaluation context for this program.

9.4 Introduction rules

In a second step, we give syntax to the introduction rules.

$$\begin{array}{lrcl} \text{Terms} & p, q & ::= & \ldots \mid () \mid (p, q) \mid \iota_1(p) \mid \iota_2(p) \mid \lambda a.p \mid \lambda x.p \mid (t, p) \\ \text{Evaluation contexts} & e & ::= & \ldots \mid [\,] \mid \pi_1[e] \mid \pi_2[e] \mid [e|e'] \mid p \cdot e \mid t \cdot e \mid \lambda x.e \end{array}$$

$$\bot_L \ \frac{}{\Gamma; [\,] : \bot \vdash \Delta} \qquad \top_R \ \frac{}{\Gamma \vdash () : \top; \Delta}$$

$$\wedge_L^1 \ \frac{\Gamma; e : A_1 \vdash \Delta}{\Gamma; \pi_1[e] : A_1 \wedge A_2 \vdash \Delta} \qquad \wedge_L^2 \ \frac{\Gamma; e : A_2 \vdash \Delta}{\Gamma; \pi_2[e] : A_1 \wedge A_2 \vdash \Delta} \qquad \wedge_R \ \frac{\Gamma \vdash p_1 : A_1; \Delta \qquad \vdash p_2 : A_2; \Delta}{\Gamma \vdash (p_1, p_2) : A_1 \Rightarrow A_2}$$

$$\vee_L \ \frac{\Gamma; e_1 : A_1 \vdash \qquad \Gamma; e_2 : A_2 \vdash \Delta}{\Gamma; [e_1|e_2] : A_1 \vee A_2 \vdash \Delta} \qquad \vee_R^1 \ \frac{\Gamma \vdash p : A_1; \Delta}{\Gamma \vdash \iota_1(p) : A \vee B} \qquad \vee_R^2 \ \frac{\Gamma \vdash p : A_2; \Delta}{\Gamma \vdash \iota_2(p) : A \vee B}$$

$$\Rightarrow_L \ \frac{\Gamma \vdash p : A; \Delta \qquad \Gamma; e : B \vdash \Delta}{\Gamma; p \cdot e : A \Rightarrow B \vdash \Delta} \qquad \Rightarrow_R \ \frac{\Gamma, a : A \vdash p : B; \Delta}{\Gamma \vdash \lambda a.p : A \Rightarrow B; \Delta}$$

$$\forall_L \ \frac{\Gamma; e : A[t/x] \vdash \Delta}{\Gamma; t \cdot e : \forall x\, A \vdash \Delta} \qquad \forall_R \ \frac{\Gamma \vdash p : A; \Delta \qquad x \text{ fresh}}{\Gamma \vdash \lambda x.p : \forall x\, A; \Delta}$$

$$\exists_L \ \frac{\Gamma; e : A \vdash \Delta \qquad x \text{ fresh}}{\Gamma; \lambda x.e : \exists x\, A \vdash \Delta} \qquad \exists_L \ \frac{\Gamma \vdash p : A[t/x]; \Delta}{\Gamma \vdash (t, p) : \exists x\, A; \Delta}$$

9.5 Sequent calculus in sequent-free style

Interestingly, distinguishing three kinds of sequents allows to represent sequent calculus proofs as a tree of formulae, not using sequents anymore, as Gentzen was representing natural deduction proofs. To reflect the three kinds of sequents, we however need to annotate formula with an information telling if it is asserted or refuted and we reuse for that the $\vdash$ symbol ($\vdash A$ to assert A and $A \vdash$ to refute A - nice alternative notations includes e.g. A and $\overline{A}$ respectively). We also need a notation for no formula, intuitively a contradiction, and we choose here the notation $\vdash$ (a nice alternative notation could be $\perp\!\!\!\perp$ where the doubled $\perp$ highlights the fact that this is a contradictory judgement, not the assertion of a false formula).

The possibility for such an alternative presentation definitely buries the historical accident that systems made from left introductions and right introductions were called sequent calculi by opposition to natural deduction which did not use sequent. All systems (natural deduction, Frege-Hilbert systems, sequent calculus) can be presented using sequents or using only formulae. This is a presentational issue independent of the structure of the calculus itself.

In a tree of formulas, the leaves are either of the form $\vdash A$ or of the form $A \vdash$. These leaves can possibly be overlined. In the following core rules, the overlines mean that a selection of leaves of the form $\vdash A$ or $A \vdash$ in the derivation are "discharged" by the rule.

$$\tilde{\mu} \ \frac{\begin{array}{c}\overline{\vdash A} \\ \vdots \\ \vdash\end{array}}{A \vdash} \qquad \mu \ \frac{\begin{array}{c}\overline{A \vdash} \\ \vdots \\ \vdash\end{array}}{\vdash A}$$

$$Cut \ \frac{\vdash A \qquad A \vdash}{\vdash}$$

The introduction rules for the main connectives are:

$$\bot_L \ \frac{}{\bot \vdash} \qquad \top_R \ \frac{}{\vdash \top}$$

$$\wedge^1_L \ \frac{A \vdash}{A \wedge B \vdash} \qquad \wedge^2_L \ \frac{B \vdash}{A \wedge B \vdash} \qquad \wedge_R \ \frac{\vdash A \qquad \vdash B}{\vdash A \Rightarrow B}$$

$$\vee_L \ \frac{A \vdash \qquad B \vdash}{A \vee B \vdash} \qquad \vee^1_R \ \frac{\vdash A}{\vdash A \vee B} \qquad \vee^2_R \ \frac{\vdash B}{\vdash A \vee B}$$

$$\Rightarrow_L \ \frac{\vdash A \qquad B \vdash}{A \Rightarrow B \vdash} \qquad \Rightarrow_R \ \frac{\begin{array}{c}\overline{\vdash A} \\ \vdots \\ \vdash B\end{array}}{\vdash A \Rightarrow B}$$

$$\forall_L \ \frac{A[t/x] \vdash}{\forall x\, A \vdash} \qquad \forall_R \ \frac{\vdash A \qquad x \text{ fresh}}{\vdash \forall x\, A}$$

$$\exists_L \ \frac{A \vdash \qquad x \text{ fresh}}{\exists x\, A \vdash} \qquad \exists_L \ \frac{\vdash A[t/x]}{\vdash \exists x\, A}$$

Of course, proof annotations could be used (using notations such as $\vdash p : A$, $e : A \vdash$, and, e.g., simply c). Note also that there is no canonical way to add annotations when using the above presentation. Indeed, if two rules are building, say $\vdash A$, then the information of which of them are bracketing which of the potential leaves $A \vdash$ is lost.

9.6 Computing with sequent calculus

Core sequent calculus can be equipped with the following simple reduction rules:

$$\begin{array}{lcl} \langle \mu\alpha.c \parallel e \rangle & \to & c[e/\alpha] \\ \langle p \parallel \tilde{\mu}x.c \rangle & \to & c[p/x] \end{array}$$

These rules are reduction rules for commands. Indeed, commands are interactions between a program and an evaluation context and it is where computation happens.

The first rule says that $\mu\alpha.c$ is a program which captures its evaluation context. The second rule says that $\tilde{\mu}x.c$ is an evaluation context which captures the program filling the context, so as to continue with another computation.

Intuitively $\tilde{\mu}x.c$ can be seen as a context of the form **let** $x = [\,]$ **in** c, i.e. as a context which wants to capture the term in its holes before continuing with c.

The above reduction system is symmetric with respect to the left-hand and right-hand side of sequents. This means in particular that there is a critical pair:

$$\begin{array}{ccc} & \langle \mu\alpha.c \parallel \tilde{\mu}x.c' \rangle & \\ \swarrow (\mu) & & (\tilde{\mu}) \searrow \\ c[\tilde{\mu}x.c'/\alpha] & & c'[\mu\alpha.c/x] \end{array}$$

It happens that this critical pair exactly reflects the dilemma between call-by-name and call-by-value reduction, with a call-by-name vs call-by-value terminology defined from the view point of the evaluation context. If the evaluation context $\tilde{\mu}x.c$ wants its argument to be evaluated first, it gives precedence to the rule (μ) and this defines call-by-value. If the evaluation context instead immediately wants to binds its argument to the name x, without prior evaluation of the argument, this is a call-by-name evaluation.

Each connective comes with its reduction rules too:

$$
\begin{array}{lll}
(\Rightarrow) & \langle \lambda a.p \parallel q \cdot e \rangle & \overset{h}{\to} \quad \left\langle \mu\alpha.\left\langle q \parallel \tilde{\mu}a.\langle p \parallel \alpha \rangle \right\rangle \middle\| e \right\rangle \\
(\wedge^1) & \langle (p_1, p_2) \parallel \pi_1[e] \rangle & \overset{h}{\to} \quad \langle p_1 \parallel e \rangle \\
(\wedge^2) & \langle (p_1, p_2) \parallel \pi_2[e] \rangle & \overset{h}{\to} \quad \langle p_2 \parallel e \rangle \\
(\vee^1) & \langle \iota_1(p) \parallel [E_1, E_2] \rangle & \overset{h}{\to} \quad \langle p \parallel E_1 \rangle \\
(\vee^2) & \langle \iota_2(p) \parallel [E_1, E_2] \rangle & \overset{h}{\to} \quad \langle p \parallel E_2 \rangle \\
(\forall) & \langle \lambda x.p \parallel t \cdot e \rangle & \overset{h}{\to} \quad \langle p[t/x] \parallel e \rangle \\
(\exists) & \langle (t, p) \parallel \lambda x.e \rangle & \overset{h}{\to} \quad \langle p \parallel e[t/x] \rangle
\end{array}
$$

9.7 Historical notes

For the purpose of easing higher-order unification, Cervesato and Pfenning designed the spine calculus [CP97] which can be seen as a term assignment of sequent calculus similar to [Her95a].

Part II

On the relations between syntax and semantics

Chapter 10

Second-order arithmetic as a meta-language

In the next chapter, we shall study soundness and completeness from a computational point of view. To do so, we place ourselves in a formal meta-logic implementing second-order arithmetic.

We consider a multi-sorted second-order arithmetic with arbitrary recursive types, so that it features not only natural numbers but also lists, tuples, as well as basic types for defining logics such as formulae, contexts, provability, etc.

quantification over all functions in finite types, i.e. with quantification over functions in System T, as well as quantification over predicates overs functions in finite types. We take as atoms all primitive recursive predicates.

Chapter 11

Tarski semantics and completeness

Chapter 12

Kripke semantics

While it is traditional to associate classical logic with Tarski models, a common variant of models for intuitionistic logic is Kripke models. A Kripke model is a family of Tarki Kripke models ordered by some notion of "time". We shall successively describe Kripke models for propositional logic, predicate logic, second-order propositional logic (i.e. System F) and System F_ω.

12.1 Kripke semantics for propositional logic

A Kripke model for propositional calculus $(\mathcal{K}, \mathcal{W}, \leq, \Vdash)$ is defined as family of Tarki interpretations depending on a set of *worlds* $w \in \mathcal{W}$, ordered by $\leq$. The relation $w \Vdash_X$, read X is *forced* at w relates worlds and atoms of the language of propositional logic: for each world w, the atoms X such as $w \vdash_X$ holds defines a propositional Tarski interpretation.

The interpretation of a formula in a Kripke model is different from the one in a Tarski model. We inductively extend the forcing relation on atoms to all formulae by:

$$\begin{array}{lcl} w \Vdash X & \triangleq & w \Vdash_X \\ w \Vdash A \Rightarrow B & \triangleq & \text{for all } w' \geq w,\ w' \Vdash A \text{ implies } w' \Vdash B \\ w \Vdash A \wedge B & \triangleq & w \Vdash A \text{ and } w \Vdash B \\ w \Vdash A \vee B & \triangleq & w \Vdash A \text{ or } w \Vdash B \end{array}$$

We also canonically extend the forcing relation to contexts:

$$\begin{array}{lcl} w \Vdash \emptyset & \triangleq & \emptyset \\ w \Vdash \Gamma, a : A & \triangleq & (w \Vdash \Gamma) \text{ and } (w \Vdash A) \end{array}$$

and to theories:

$$w \Vdash \mathcal{T} \quad \triangleq \quad \text{for all } A \in \mathcal{T},\ w \Vdash A$$

This is enough to define Kripke validity, which is truth in all worlds at all models. We give two definitions, one for finite contexts, and the other for theories:

$$\begin{array}{lcl} \Gamma \Vdash A & \triangleq & \text{for all } \mathcal{K} = (\mathcal{W}, \leq, \Vdash), \text{ for all } w \in \mathcal{W},\ w \Vdash \Gamma \text{ implies } w \Vdash A \\ \mathcal{T} \Vdash A & \triangleq & \text{for all } \mathcal{K} = (\mathcal{W}, \leq, \Vdash), \text{ for all } w \in \mathcal{W},\ w \Vdash \mathcal{T} \text{ implies } w \Vdash A \end{array}$$

12.2 Soundness and completeness of Kripke models with respect to intuitionistic provability

The main interest of Kripke models is that it exactly captures intuitionistic provability in that it is a sound and complete semantics for intuitionistic provability:

Theorem 24 (Soundness of Kripke models for intuitionistic provability, standard presentation)
$\Gamma \vdash_I A$ implies $\Gamma \Vdash A$, and, more generally, $\mathcal{T} \vdash_I A$ implies $\mathcal{T} \Vdash A$.

Theorem 25 (Completeness of Kripke models for intuitionistic provability, standard presentation)
$\Gamma \Vdash A$ implies $\Gamma \vdash_I A$, and, more generally, $\mathcal{T} \Vdash A$ implies $\mathcal{T} \vdash_I A$.

We shall give the proof of soundness and completeness as programs

Bibliography

[And92] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Log. Comput.*, 2(3):297–347, 1992.

[BHF01] Kensuke Baba, Sachio Hirokawa, and Ken-etsu Fujita. Parallel reduction in type free $\lambda\mu$-calculus. *Electronic Notes in Theoretical Computer Science*, 42:52–66, 2001.

[BS91] Ulrich Berger and Helmut Schwichtenberg. An inverse of the evaluation functional for typed lambda-calculus. In *Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science, 15-18 July, 1991, Amsterdam, The Netherlands*, pages 203–211. IEEE Computer Society, 1991.

[CFC58] Haskell B. Curry, Robert Feys, and William Craig. *Combinatory Logic*, volume 1. North-Holland, 1958. §9E.

[CH98] Pierre-Louis Curien and Hugo Herbelin. Computing with abstract Böhm trees. In Masahiko Sato and Yoshihito Toyama, editors, *Fuji International Symposium on Functional and Logic Programming (FLOPS '98), Kyoto, Japan, April 2-4, 1998*, pages 20–39. World Scientific, Singapore, 1998.

[CH09] Felice Cardone and J. Roger Hindley. History of lambda-calculus and combinatory logic. In D. M. Gabbay and J. Woods, editors, *Handbook of the History of Logic, Logic from Russell to Church*, volume 5, pages 723–817. Elsevier, Amsterdam, 2009.

[CP97] Iliano Cervesato and Frank Pfenning. Linear higher-order pre-unification. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 422–433. IEEE Computer Society, 1997.

[Cro99] Tristan Crolard. A confluent lambda-calculus with a catch/throw mechanism. *J. Funct. Program.*, 9(6):625–647, 1999.

[Cur34] Haskell B. Curry. Functionality in combinatory logic. *Proceedings of the National Academy of Sciences of the United States of America*, 20:584–590, 1934.

[Cur42] Haskell B. Curry. The combinatory foundations of mathematical logic. *The Journal of Symbolic Logic*, 7(2):49–64, 1942.

[dB68] Nicolaas de Bruijn. Automath, a language for mathematics. Technical Report 66-WSK-05, Technological University Eindhoven, November 1968.

[dB78] Nicolaas de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report 78-WSK-03, Technological University Eindhoven, 1978.

[DJS95] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. LKQ and LKT: sequent calculi for second order logic based upon dual linear decompositions of the classical implication. In *Advances in Linear Logic*, volume 222, pages 211–224. Cambridge University Press, 1995.

[Dra80] Albert G. Dragalin. New kinds of realizability and Markov's rule. *Soviet Mathematical Doklady*, 251:534–537, 1980.

[Fis72] Michael J. Fischer. Lambda-calculus schemata. In *Proc. ACM Conference on Proving Assertions About Programs*, volume 7(1) of *SIGPLAN Notices*, pages 104–109. ACM Press, New York, 1972.

[Fre67] Gottlob Frege. Begriffsschrift. In *From Frege to Gödel: A Source Book in Mathematical Logic*, pages 1–82. Harvard University Press, 1967. original version in German "Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinenDenkens", 1879.

[Fri78] Harvey Friedman. Classically and intuitionistically provably recursive functions. In D. S. Scott and G. H. Muller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–27. Springer, Berlin/Heidelberg, 1978.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935. English Translation in [Sza69], "Investigations into logical deduction", pages 68-131.

[Gen36] Gerhard Gentzen. Die Widerspruchfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112, 1936. English translation in [Sza69], "The consistency of elementary number theory".

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[Gir01] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[Göd32] Kurt Gödel. Zur intuitionistischen arithmetik und zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums*, 4:34–38, 1932. Reprinted in English translation as "On intuitionistic arithmetic and number theory" in The Undecidable, M. Davis, ed., pp. 75-81.

[Göd58] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3):280–287, December 1958.

[Gri90] Timothy G. Griffin. The formulae-as-types notion of control. In *Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90, San Francisco, CA, USA, 17-19 Jan 1990*, pages 47–57. ACM Press, New York, 1990.

[Her95a] Hugo Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1995.

[Her95b] Hugo Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Ph.D. thesis, University Paris 7, January 1995.

[How69] William A. Howard. The formulae-as-types notion of constructions. In *to H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1969. Published only in 1980.

[KCR+98] Richard Kelsey, (ed.), William Clinger, (ed.), Jonathan Rees, (ed.), Hal Abelson, N. I. Adams IV, D. H. Bartley, G. Brooks, R. Kent Dybvig, Daniel P. Friedman, R. Halstead, Chris Hanson, Christopher T. Haynes, Eugene Kohlbecker, D. Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy L. Steele Jr., Gerald J. Sussman, and Mitchell Wand. Revised^5 report on the algorithmic language scheme. *ACM SIGPLAN Notices*, 33(9):26–76, 1998.

[Kle45] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945.

[Kle62] Stephen C. Kleene. *Introduction to Metamathematics*. North-Holland, 1962.

[Kol67] Andrej N. Kolmogorov. On the principle of the excluded middle. In *From Frege to Gödel: A Source Book in Mathematical Logic*, pages 414–447. Harvard University Press, 1967. original version in Russian "О принципе tertium non datur" in Mat. Sbornik 32:646–667, 1925.

[KPT96] Delia Kesner, Laurence Puel, and Val Tannen. A typed pattern calculus. *Inf. Comput.*, 124(1):32–61, 1996.

[Kre51] Georg Kreisel. On the interpretation of non-finitist proofs - part i. *J. Symb. Log.*, 16(4):241–267, 1951.

[Kri90] Jean-Louis Krivine. Opérateurs de mise en mémoire et traduction de gödel. *Archive for Mathematical Logic*, 30(4):241–267, 1990.

[Kur51] Sigekatu Kuroda. Intuitionistische untersuchungen des formalistischen logik. *Kuroda Math. J.*, 3:35–47, 1951.

[Lam72] Joachim Lambek. Deductive systems and categories iii. cartesian closed categories, intuitionist propositional calculus, and combinatory logic. In F.W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 57–82. Springer Berlin Heidelberg, 1972.

[Lan65] Peter Landin. A generalisation of jumps and labels. Technical Report ECS-LFCS-88-66, UNIVAC Systems Programming Research, August 1965. Reprinted in `Higher Order and Symbolic Computation`, 11(2), 1998.

[Lau02] Olivier Laurent. *Étude de la polarisation en logique.* Ph.D. thesis, University Aix-Marseille II, March 2002.

[Lev99] Paul Blain Levy. Call-by-push-value: A subsuming paradigm. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings*, volume 1581 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 1999.

[LM09] Chuck Liang and Dale Miller. A unified sequent calculus for focused proofs. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 355–364. IEEE Computer Society, 2009.

[LRS93] Yves Lafont, Bernhard Reus, and Thomas Streicher. Continuations semantics or expressing implication by negation. Technical Report 9321, Ludwig-Maximilians-Universität, München, 1993.

[Mog88] Eugenio Moggi. Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, Edinburgh Univ., 1988.

[MT10] Paul-André Melliès and Nicolas Tabareau. Resource modalities in tensor logic. *Ann. Pure Appl. Logic*, 161(5):632–653, 2010.

[Mur90] Chetan Murthy. *Extracting constructive content from classical proofs.* Ph.D. thesis, Cornell University, 1990.

[Par91] Michel Parigot. Free deduction: An analysis of "computations" in classical logic. In Andrei Voronkov, editor, *Logic Programming, First Russian Conference on Logic Programming, Irkutsk, Russia, September 14-18, 1990 - Second Russian Conference on Logic Programming, St. Petersburg, Russia, September 11-16, 1991, Proceedings*, volume 592 of *Lecture Notes in Computer Science*, pages 361–380. Springer, 1991.

[Pel99] Francis Jeffry Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20(1):1–31, 1999.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1:125–159, 1975.

[PM68] Dag Prawitz and Per-Erik Malmnäs. A survey of some connections between classical, intuitionistic and minimal logic. In H. Arnold Schmidt, K. Schütte, and H. J. Thiele, editors, *Contributions to Mathematical Logic, Proceedings of the Logic Colloquium, Hannover 1966*, pages 215–229. North-Holland Publishing Company, 1968.

[Pra65] Dag Prawitz. *Natural Deduction, a Proof-Theoretical Study.* Almqvist and Wiksell, Stockholm, 1965.

[Rey72] John C. Reynolds. Definitional interpreters for higher-order programming languages. In *ACM '72: Proceedings of the ACM annual conference*, pages 717–740, New York, NY, USA, 1972. ACM Press.

[Rey93] John C. Reynolds. The discoveries of continuations. *LISP and Symbolic Computation*, 6(3):233–247, 1993.

[Sch24] Moses Schönfinkel. Über die bausteine der mathematischen logik. *Mathematische Annalen*, 92:305–316, 1924.

[Sel03] Jonathan P. Seldin. Curry's anticipation of the types used in programming languages. Available on the author web page, 2003.

[SS75] Gerald J. Sussman and Guy L. Steele Jr. Scheme: An interpreter for extended lambda calculus. Technical Report AIM-349, Massachusetts Institute of Technology, Cambridge, MA, USA, 1975. Reprinted in `Higher Order and Symbolic Computation`, 11(4), 1998.

[SU99] Morten Heine Sørensen and Paweł Urzyczyn. *Lectures on the Curry-Howard Isomorphism.* 1999.

[Sza69] Manfred E. Szabo, editor. *The Collected Works of Gerhard Gentzen.* North Holland, Amsterdam, 1969.

[Tai67] William W. Tait. Intensional interpretations of functionals of finite type i. *Journal of Symbolic Logic*, 32(2):198–212, 1967.

[Ter03] Terese. *Term rewriting systems*. Cambridge tracts in theoretical computer science. Cambridge University Press, Cambridge, New York, 2003.

[Tro73] Anne S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1973.

[Tro91] Anne S. Troelstra. History of constructivism in the 20th century. Technical Report ML-91-05, University of Amsterdam, 1991.

[Tro99] A. S. Troelstra. From constructivism to computer science. *Theoretical Computer Science*, 211:233–252, 1999.

[WC85] Mitchell Wand William Clinger, Daniel P. Friedman. A scheme for a higher-level semantic algebra. In *Algebraic methods in semantics: Proceedings of the US-French Seminar on the Application of Algebra to Language Definition and Compilation (Fontainebleau, France, June 82)*, pages 237–250, Cambridge, 1985. Cambridge University Press. Appeared as a Technical Report of the University of Indiana in 1983.

[Zei08] Noam Zeilberger. On the unity of duality. *Ann. Pure Appl. Logic*, 153(1-3):66–96, 2008.