FREE DEDUCTION: AN ANALYSIS OF "COMPUTATIONS" IN CLASSICAL LOGIC

Michel Parigot
Equipe de logique – CNRS UA 753
45–55 5ème étage, Université Paris 7
2 place jussieu, 75251 PARIS Cedex 05, FRANCE
e–mail: parigot@logique.jussieu.fr

Abstract: Cut–elimination is a central tool in proof–theory, but also a way of computing with proofs used for constructing new functional languages. As such it depends on the properties of the deduction system in which proofs are written.

For intuitionistic logic, natural deduction allows a cut–elimination procedure which effectively provides a computation mechanism with deep theoretical properties such as confluence and strong normalisation. For classical logic, on the contrary, neither natural deduction nor sequent calculus provide a suitable cut–elimination, and, in fact, the computational meaning of classical proofs is an open problem.

In this paper, a new deduction sytem is introduced: free deduction. Free deduction is an adequate system for classical logic, allowing a global cut–elimination procedure in the style of intuitionistic natural deduction. We prove that, provided a choice of inputs (which corresponds to a fundamental non–determinism of classical logic), the cut–elimination procedure in free deduction provides a computation mechanism for classical logic which satisfies confluence and strong normalisation.

1. INTRODUCTION

Cut–elimination is a central tool in proof–theory, but also a way of computing with proofs. As such it depends on the properties of the deduction system in which proofs are written.

In the case of intuitionistic logic, natural deduction allows a cut–elimination procedure which effectively provides a computation mechanism – widely used as a theoretical (and even implementational) base for functional programming languages. Via the so–called Curry–Howard isomorphism, it corresponds to normalisation in typed lambda–calculus. As such it has deep theoretical properties which ensure its computational pertinence: (i) confluence (or Church–Rosser property), which says that the computation mechanism is deterministic, and (ii) strong normalisation, which says that every computation terminates.

But if we turn to classical logic, the situation radically changes. Classical natural deduction is no more satisfactory from the viewpoint of cut–elimination and even the notion of cut–free proof becomes problematic. The addition of the absurdity rule conflicts with the definition of a suitable cut–elimination. Sequent calculus, on the other hand, is a deduction system for classical logic in

which cut–elimination is a powerful tool for proof–theory. But it seems difficult to assign a computational meaning to cut–elimination in sequent calculus. This is not only because the computation process fails to have good properties, but more deeply because it is not defined in a canonical way.

What are the reasons for that situation? One could simply give an a priori answer such as: this is just that classical logic is not constructive. Because I find this answer unsatisfactory, I have tried to get a more technical answer, which could explain in what precise sense it is not constructive. The first step was to separate the problems due to classical logic from the ones due to the deduction systems in which classical proofs are written. This led me to the definition of a new deduction system for classical logic: free deduction (FD).

Free deduction is a fully symetric system (in the style of sequent calculus) in which formulas and sequents are the usual ones, and the notion of cut internalised (as in natural deduction). It enjoys the cut–elimination theorem, and could be used as an alternative system for proof–theoretic studies. Sequent calculus and natural deduction are obtained from free deduction using a very natural and uniform restriction: it suffices to kill certain premises of the rules (they are not mentionned, but have the effect of axioms). In this respect, FD appears both as a system and a meta–system in which one can transform proofs from one system to another.

But the main distinctive properties of free deduction concern the computational aspects. Two cut–elimination procedures are available in FD: a local procedure (as in sequent calculus) which proceeds by permutation of rules and a global one (as in natural deduction) which proceeds by composition of proofs. The existence of global cut–elimination procedure is a crucial property of FD: it effectively provides a computation mechanism for classical logic, for which confluence and strong normalisation hold. A confluence result may seem strange, because of the non–deterministic nature of classical logic. But in fact, non–determinism is entirely included in the choice of the inputs: one has to choose whether the input of a formula is to the right or to the left of the sequents.

Looking from the point of view of FD, one obtains a good explanation of the computational differences between natural deduction and sequent calculus. In these systems, the choice of the inputs is syntactically imposed by the choice of the killed premisses of the rules. In (intuitionistic) natural deduction inputs are systematically to the left (in a sequent $A_1,...,A_n \vdash B$, the formulas $A_1,...,A_n$ are inputs and B is an output) and this leads to a deterministic computation mechanism (the usual functional interpretation of intuitionistic proofs). In sequent calculus, inputs are not always available, and this prevents to define a suitable computational mechanism.

In free deduction, all the inputs are available. As a consequence cut–elimination provides a non–deterministic computation mechanism for classical logic. But we prove that the fundamental non–determinism of classical logic amounts to a choice of the inputs: each such choice give birth to a deterministic computation mechanism satisfying confluence and strong normalisation (note that for us, determinism refers to the result and not to the computation process, otherwise confluence would have no meaning).

In what follows we only discuss propositional free deduction, but the system and the results obviously generalise to the predicate case.

Notations: latin letters A,B,C,.. denote formulas and greek letters Γ,Δ,Π,Σ denote sets of formulas;

2. THE SYSTEM OF FREE DEDUCTION

2.1 Rules of free deduction.

Axiom

$$A \vdash A$$

Conjunction

$$\frac{\Gamma, A \wedge B \vdash \Delta \qquad \Pi \vdash A, \Sigma \qquad \Pi' \vdash B, \Sigma'}{\Gamma, \Pi, \Pi' \vdash \Delta, \Sigma, \Sigma'} \qquad \frac{\Gamma \vdash A \wedge B, \Delta \qquad \Pi, A \vdash \Sigma \quad \{\Pi, B \vdash \Sigma\}}{\Gamma, \Pi \vdash \Delta, \Sigma}$$

Disjunction

$$\frac{\Gamma, A \vee B \vdash \Delta \qquad \Pi \vdash A, \Sigma \quad \{\Pi \vdash B, \Sigma\}}{\Gamma, \Pi \vdash \Delta, \Sigma} \qquad \frac{\Gamma \vdash A \vee B, \Delta \qquad \Pi, A \vdash \Sigma \qquad \Pi', B \vdash \Sigma'}{\Gamma, \Pi, \Pi' \vdash \Delta, \Sigma, \Sigma'}$$

Negation

$$\frac{\Gamma, \neg A \vdash \Delta \qquad \Pi, A \vdash \Sigma}{\Gamma, \Pi \vdash \Delta, \Sigma} \qquad \frac{\Gamma \vdash \neg A, \Delta \qquad \Pi \vdash A, \Sigma}{\Gamma, \Pi \vdash \Delta, \Sigma}$$

Implication

$$\frac{\Gamma, A \neg B \vdash \Delta \qquad \Pi, A \vdash \Sigma \quad \{\Pi \vdash B, \Sigma\}}{\Gamma, \Pi \vdash \Delta, \Sigma} \qquad \frac{\Gamma \vdash A \neg B, \Delta \qquad \Pi \vdash A, \Sigma \qquad \Pi', B \vdash \Sigma'}{\Gamma, \Pi, \Pi' \vdash \Delta, \Sigma, \Sigma'}$$

A sequent between brackets beside a rule means that there are in fact two rules, the one which is written and the one formed by replacing the right premise by the sequent between brackets.

The left premise of a rule is called the main premise and the other ones, secondary premises; the formula mentionned in the premise is called the active formula of the premise and the other ones form the context; a rule whose main premise has A as active formula is called an elimination of A.

Meaning of the rules.

Sequents are interpreted as usual: the sequent $A_1,...,A_n \vdash B_1,...,B_m$ is true if and only if the formula $A_1 \wedge ... \wedge A_n \rightarrow B_1 \vee ... \vee B_m$ is true; in particular $\vdash A$ (resp. $A \vdash$) is true if and only if A is true (resp. false).

The intuitive meaning of the rules is that the premises give the conditions to derive a contradiction; for instance for the connective and:

– one derives a contradiction from "A ∧ B false" and "A true and B true";

– one derives a contradiction from "A ∧ B true" and "A false" (or "B false").

Structural rules.

There are two ways of adding structural rules to FD :

(i) Explicit (or local) structural rules: weakening and contraction are governed by their usual rules.

(ii) Implicit (or global) structural rules:

– weakening is obtained (as in natural deduction) by the convention that the active formula of a premise does not necessary occur; when it occurs the premise is called strict;

– contraction is obtained by the convention that the contexts in the premises of a rule are always contracted in the conclusion of the rule.

As usual, a more complex structure is in fact needed from an algorithmic viewpoint: one needs indexed formulas A^x (or x:A in the notation of typed lambda–calculus) and not only formulas.

Explicit structural rules are more convenient for the local cut–elimination procedure and implicit structural rules are more convenient for the global one.

Proposition (completeness) FD is complete for provability in classical logic. Moreover,

(i) there is a "direct" translation of (cut–free) proofs of sequent calculus into (cut–free) proofs in this system (which don't use additional structural rules);

(ii) there is a "direct" translation of proofs of natural deduction into proofs in this system, (which respects the structure of cuts).

Remark. Like natural deduction, free deduction can be presented both as a deduction system of sequents and a deduction system of formulas.

Variant. A important variant FD' of the system FD (which is not considered in this paper) is obtained by "unifying" the pairs of rules for $\wedge d$, $\vee g$ and $\neg g$ using secondary premises with two active formulas:

$$\frac{\Gamma \vdash A \wedge B,\ \Delta \qquad \Pi,\ A,\ B \vdash \Sigma}{\Gamma,\ \Pi \vdash \Delta,\ \Sigma}$$

$$\frac{\Gamma,\ A \vee B \vdash \Delta \qquad \Pi \vdash A,\ B,\ \Sigma}{\Gamma,\ \Pi \vdash \Delta,\ \Sigma}$$

$$\frac{\Gamma,\ A \neg B \vdash \Delta \qquad \Pi,\ A \vdash B,\ \Sigma}{\Gamma,\ \Pi \vdash \Delta,\ \Sigma}$$

In FD' binary connectives become symetric, from the algorithmic point of view.

2.2 Formula property.

The formula property is a very simple property, which is in fact the base of the global cut–elimination procedure and therefore of the main computational properties of free deduction. It says that each formula occuring in the conclusion of a proof comes from an axiom (in the case of explicit strutural rules one needs to add "or from a weakening"). This has to be compared with

natural deduction where we have only a left formula property: each formula occuring to the left of the sequent conclusion of a proof comes from an axiom.

2.3 Permutability property.

The permutability property is the base of the local cut–elimination property: it is a systematization of a partly existing property of sequent calculus.

Permutability property between logical rules.

The rules of FD are permutable in the following sense: suppose that we have a rule R' whose conclusion is a premise of a rule R and C_1 is the active formula of this premise; then we can permute the two rules in the following way: first apply R to the premise of R' containing C_1 and then R' to the conclusion; the conclusion of the proof is unchanged (in fact the only sequent which is changed is the intermediate sequent between the two rules) and no structural rule is added. Moreover the permutation process is reversible: if one permutes R and R' and then R' and R, we get the original proof.

Here is the general scheme of permutation (in order to cover the general situation we represent sequents by lists of formulas, indicate active formulas in boldface and consider the premises of a rule as unordered):

$$\dfrac{\dfrac{\Gamma_1', C_1, C_1' \qquad \Gamma_2', C_2' \qquad \Gamma_3', C_3'}{\Gamma_1', \Gamma_2', \Gamma_3', C_1 \qquad\qquad \Gamma_2, C_2 \qquad \Gamma_3, C_3}\ R'}{\Gamma_1', \Gamma_2', \Gamma_3', \Gamma_2, \Gamma_3}\ R$$

becomes

$$\dfrac{\dfrac{\Gamma_1', C_1', C_1 \qquad \Gamma_2, C_2 \qquad \Gamma_3, C_3}{\Gamma_1', \Gamma_2, \Gamma_3, C_1' \qquad\qquad \Gamma_2', C_2' \qquad \Gamma_3', C_3'}\ R}{\Gamma_1', \Gamma_2, \Gamma_3, \Gamma_2', \Gamma_3'}\ R'$$

Permutability property for contractions.

The permutation of a contraction and a logical rule is not reversible: the permutation is only possible if the contraction precedes the logical rule (otherwise the contracted occurences can come from different premises of the logical rule, and the permutation is impossible). There are two different cases: if the contracted formula is not an active formula of the logical rule, then the permutation is obvious and doesn't change the structure of the proof; if the contracted formula is an active formula of the logical rule, then the permutation produces a duplication of the logical rule:

$$\dfrac{\dfrac{\Gamma_1, C_1, C_1}{\Gamma_1, C_1}\ Ct \qquad\qquad \Gamma_2, C_2 \qquad \Gamma_3, C_3}{\Gamma_1, \Gamma_2, \Gamma_3}\ R$$

becomes

$$\dfrac{\dfrac{\dfrac{\Gamma_1, C_1, C_1 \qquad \Gamma_2, C_2 \qquad \Gamma_3, C_3}{\Gamma_1, C_1, \Gamma_2, \Gamma_3}\,R \qquad \Gamma_2, C_2 \qquad \Gamma_3, C_3}{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_2, \Gamma_3}\,R}{\Gamma_1, \Gamma_2, \Gamma_3}\,Ct$$

Permutability property for weakenings.

The permutation of a weakening and a logical rule is interesting only if the weakening precedes the logical rule (otherwise the permutation is possible but arbitrary: one has to choose an arbitrary premise of the logical rule). There are two different cases: if the weakened formula is not an active formula of the logical rule, then the permutation is obvious and doesn't change the structure of the proof; if the weakened formula is an active formula of the logical rule, then the permutation produces an erasing of the logical rule:

$$\dfrac{\dfrac{\Gamma_1}{\Gamma_1, C_1}\,Wk \qquad \Gamma_2, C_2 \qquad \Gamma_3, C_3}{\Gamma_1, \Gamma_2, \Gamma_3}\,R$$

becomes

$$\dfrac{\Gamma_1}{\Gamma_1, \Gamma_2, \Gamma_3}\,Wk$$

3. LOGICAL PROPERTIES OF FREE DEDUCTION

3.1 Cuts and their elimination.

A cut in a proof is either (i) an elimination R of a weakened formula A, or (ii) a right (resp. left) elimination R of a formula A, such that the proof of the main premise of R contains a left (resp. right) elimination of the same formula A ("the same" means that the two occurences of A are related by an axiom); in this case, A is called a cut–formula.

Example. $A \wedge B$ is a cut–formula in the following proof:

$$\dfrac{\dfrac{\overset{d_0}{\Gamma,\ A\wedge B \vdash A\wedge B,\ \Delta} \qquad \overset{d_1}{\Pi \vdash A,\ \Sigma} \qquad \overset{d_2}{\Pi' \vdash B,\ \Sigma'}}{\Gamma,\ \Pi,\ \Pi' \vdash A\wedge B,\ \Delta, \Sigma, \Sigma'}\,\wedge l \qquad \overset{d_3}{\Pi'',\ A \vdash \Sigma''}}{\Gamma,\ \Pi,\ \Pi',\ \Pi'' \vdash \Delta,\ \Sigma,\ \Sigma',\ \Sigma''}\,\wedge r$$

This notion of cut enjoys the usual proof–theoretic properties.

Subformula property: each formula which appears in a proof without cut is a subformula of a formula of the conclusion of the proof.

Cut elimination theorem: each proof can be effectively transformed into a proof without cut.

The cut–elimination theorem is proved using a local procedure inspired by Gentzen procedure for sequent calculus (cf [2]). Because of the permutation property, it can be presented more uniformly.

Here is an example of a proof without cuts which is neither in natural deduction, nor in sequent calculus:

$$\dfrac{\dfrac{\dfrac{A\vee\neg A \vdash A\vee\neg A \qquad \neg A \vdash \neg A}{\neg A \vdash A\vee\neg A}\,\vee l \qquad \dfrac{A\vee\neg A \vdash A\vee\neg A \qquad A \vdash A}{A \vdash A\vee\neg A}\,\vee l}{\vdash A\vee\neg A}\,\neg r$$

3.2 Interpretation of sequent calculus

There is a direct interpretation of (cut–free) sequent calculus into free deduction. Axioms (and structural rules) are unchanged and logical rules of sequent calculus appear as particular cases of the corresponding rules of free deduction: it suffices to kill the main premise (i.e. the main premise do not appear in the rule, but has the effect of an axiom). Note that one would obtain a different version of sequent calculus starting from the alternative system FD' instead of FD.
The killed premises are written inside brackets.

$$\dfrac{[A\wedge B \vdash A\wedge B] \qquad \Pi \vdash A, \Sigma \qquad \Pi' \vdash B, \Sigma'}{\Pi, \Pi' \vdash A\wedge B, \Sigma, \Sigma'} \qquad \dfrac{[A\wedge B \vdash A\wedge B] \qquad \Pi, A \vdash \Sigma \quad \{\Pi, A \vdash \Sigma\}}{\Pi, A\wedge B \vdash \Sigma}$$

$$\dfrac{[A\vee B \vdash A\vee B] \qquad \Pi \vdash A, \Sigma \quad \{\Pi \vdash B, \Sigma\}}{\Pi \vdash A\vee B, \Sigma} \qquad \dfrac{[A\vee B \vdash A\vee B] \qquad \Pi, A \vdash \Sigma \qquad \Pi', B \vdash \Sigma'}{\Pi, \Pi', A\vee B \vdash \Sigma, \Sigma'}$$

$$\dfrac{[\neg A \vdash \neg A] \qquad \Pi, A \vdash \Sigma}{\Pi \vdash \neg A, \Sigma} \qquad \dfrac{[\neg A \vdash \neg A] \qquad \Pi \vdash A, \Sigma}{\Pi, \neg A \vdash \Sigma}$$

$$\dfrac{[A\to B \vdash A\to B] \qquad \Pi, A \vdash \Sigma \quad \{\Pi \vdash B, \Sigma\}}{\Pi \vdash A\to B, \Sigma} \qquad \dfrac{[A\to B \vdash A\to B] \qquad \Pi \vdash A, \Sigma \qquad \Pi', B \vdash \Sigma'}{\Pi, \Pi', A\to B \vdash \Sigma, \Sigma'}$$

The following proposition shows that free deduction can be seen as a meta–system in which proofs can be transformed internally (by permutation of rules) from one system to another.

Proposition. There is a procedure which transform each (cut–free) proof of free deduction into a (cut–free) proof of sequent calculus without cut.

The procedure is based on the permutation of rules whose main premise is not an axiom; another possibility is the global elimination of structural cuts of section 4.3.

3.3 Interpretation of natural deduction

Rules of natural deduction also appear as particular cases of rules of free deduction. One crucial property of natural deduction is to have inputs to the left. It can be expressed by a killing of (i) the main premise, for left rules, and (ii) the secondary premises with a left active formula, for right rules. The resulting system is a natural deduction system with multiple conclusions.

$$\frac{[A\wedge B \vdash A\wedge B] \quad \Pi \vdash A, \Sigma \quad \Pi' \vdash B, \Sigma'}{\Pi, \Pi' \vdash A\wedge B, \Sigma, \Sigma'} \qquad \frac{\Gamma \vdash A\wedge B, \Delta \quad [A \vdash A] \quad \{[B \vdash B]\}}{\Gamma \vdash A, \Delta}$$

$$\frac{[A\vee B \vdash A\vee B] \quad \Pi \vdash A, \Sigma \quad \{\Pi \vdash B, \Sigma\}}{\Pi \vdash A\vee B, \Sigma} \qquad \frac{\Gamma \vdash A\vee B, \Delta \quad [A \vdash A] \quad [B \vdash B]}{\Gamma \vdash A, B, \Delta}$$

$$\frac{[\neg A \vdash \neg A] \quad \Pi, A \vdash \Sigma}{\Pi \vdash \neg A, \Sigma} \qquad \frac{\Gamma \vdash \neg A, \Delta \quad \Pi \vdash A, \Sigma}{\Gamma, \Pi \vdash \Delta, \Sigma}$$

$$\frac{[A\to B \vdash A\to B] \quad \Pi, A \vdash \Sigma \quad \{\Pi \vdash B, \Sigma\}}{\Pi \vdash A\to B, \Sigma} \qquad \frac{\Gamma \vdash A\to B, \Delta \quad \Pi \vdash A, \Sigma \quad [B \vdash B]}{\Gamma, \Pi \vdash B, \Delta, \Sigma}$$

For the usual natural deduction system, there is one more requirement, which corresponds to the functional interpretation of natural deduction: the ouput must be unique or, in other words, sequents must have at most one formula to the right. It can be also obtained by a killing of premises, starting from a variant of FD with at most one formula to the right (the restriction of FD to sequents with at most one formula to the right is complete for classical logic). Moreover in order to keep completeness with respect to classical logic one additional rule is needed: the absurdity rule. It is obtained from left negation rule by killing the secondary premise, instead of the main premise (thus destroying the symetry).

4. COMPUTATIONAL PROPERTIES OF FREE DEDUCTION

In this section we define and study a global procedure for cut–elimination in free deduction. In fact, this procedure does more than cut elimination. Because free deduction contains cut–free subsystems of different natures (like sequent calculus and natural deduction) which are complete for classical logic, one has the possibility to compute not only cut–free proofs, but cut–free proofs of a particular form. For doing this, one has to define generalised notions of cuts including not only logical cuts, but also structural cuts depending on the particular form choosen.

In what follows we investigate the case where the particular form is a cut–free proof of sequent calculus. In this case a generalised cut is a rule whose main premise is not a strict axiom.

4.1 Composition of proofs.

The global cut–elimination procedure uses implicit structural rules. We therefore deal with indexed formulas A^x, A^y, B^x, B^y... (indexes are mentionned only when necessary). In sequents, left and right occurences receive distinct indexes. A bounded occurence of A^x in a proof d is an occurence of A^x in a subproof of a premise of a rule where A^x is active; a free occurence of A^x in d, is an occurence which is not bounded. We suppose for simplicity that an active formula A^x of a sequent in a proof never occurs outside the subproof of this sequent. A formula is free (resp. bounded) in a proof, if it has a free (resp. bounded) occurence in this proof.

The basic mechanism of the global cut–elimination procedure is the composition of proofs, as for intuitionistic natural deduction or typed lambda–calculus: one replaces in a proof d, occurences of a hypothesis A, by a proof e of A. In natural deduction inputs are necessary to the left of sequents and therefore this is the only way of composing proofs; in free deduction one has the freedom to decide, for each formula, whether the input is to the right or to the left and this gives two possible ways of composing proofs:

input to the left: $d[[\vdash A^z]e/A^x\vdash]$

input to the right: $d[[A^z\vdash]e/\vdash A^x]$.

For A^x not bounded in d, the proof $d[[\vdash A^z]e/A^x\vdash]$ (resp. $d[[A^z\vdash]e/\vdash A^x]$) is defined as the result of replacing in d, each axiom $A^x \vdash A^y$ (resp. $A^y \vdash A^x$) by the proof $e[A^y/A^z]$, where $e[A^y/A^z]$ is defined, as usual, as the result of replacing the formula A^z by the formula A^y in the proof e (a formal definition of substitution is given in § 5.1). Note that because of our conventions on indexes one could avoid to mention explicitely left and right occurences, but we would loose clarity.

4.2 Logical cuts.

A logical cut is a right (resp. left) elimination of a non contracted formula A^x, whose main premise is the conclusion of a left (resp. right) elimination of A^y, whose main premise is the axiom $A^y \vdash A^x$ (resp $A^x \vdash A^y$).

Each connective has an associated cut. Let us examine the cases of negation and conjunction.

negation

$$\dfrac{\dfrac{\neg A \vdash \neg A \qquad \begin{matrix}d_1\\ \vdots\\ \Pi,\ A^x \vdash \Sigma\end{matrix}}{\Pi \vdash \neg A,\ \Sigma}\ \neg l \qquad \begin{matrix}d_2\\ \vdots\\ \Pi' \vdash A^y,\ \Sigma'\end{matrix}}{\Pi,\ \Pi' \vdash \Sigma,\ \Sigma'}\ \neg r$$

The natural way to eliminate this cut is to replace the proof by its reductions, which is obtained, as in natural deduction, by choosing a **left input** for A:

$$\begin{matrix}d_1[[\vdash A^y]d_2/A^x\vdash]\\ \vdots\\ \Pi,\ \Pi'' \vdash \Sigma,\ \Sigma''\end{matrix}$$

where $\Pi'' = \Pi'$ and $\Sigma'' = \Sigma'$ if $A^x$is not weakened in d_1, or $\Pi'' = \Sigma'' = \emptyset$ otherwise.
But there is another possible reducts, corresponding to the choice of a **right input** for A:

$$\begin{matrix}d_2[[A^x\vdash]d_1/\vdash A^y]\\ \vdots\\ \Pi',\ \Pi'' \vdash \Sigma',\ \Sigma''\end{matrix}$$

where $\Pi'' = \Pi$ and $\Sigma'' = \Sigma$ if A^y is not weakened in d_2, or $\Pi'' = \Sigma'' = \emptyset$ otherwise.
Because of the formula property, these reducts are well defined, but they are in general completely different. In the existence of these two possibilities, lies the fundamental computational non–determinism of classical logic.

conjunction

$$\dfrac{\dfrac{A\wedge B \vdash A\wedge B \qquad \begin{matrix}d_1\\ \vdots\\ \Pi \vdash A^y,\ \Sigma\end{matrix} \qquad \begin{matrix}d_2\\ \vdots\\ \Pi' \vdash B,\ \Sigma'\end{matrix}}{\Pi,\ \Pi' \vdash A\wedge B,\ \Sigma,\ \Sigma'}\ \wedge l \qquad \begin{matrix}d_3\\ \vdots\\ \Pi'',\ A^x \vdash \Sigma''\end{matrix}}{\Pi,\ \Pi',\ \Pi'' \vdash \Sigma,\ \Sigma',\ \Sigma''}\ \wedge r$$

One has again two possible reductions:

$$\begin{matrix}d_3[[\vdash A^y]d_1/A^x\vdash]\\ \vdots\\ \Pi_1,\ \Pi'' \vdash \Sigma_1,\ \Sigma''\end{matrix} \quad \text{or} \quad \begin{matrix}d_1[[A^x\vdash]d_3/\vdash A^y]\\ \vdots\\ \Pi,\ \Pi_2 \vdash \Sigma,\ \Sigma_2\end{matrix}$$

where Π_1,Σ_1 are respectively Π,Σ or $\emptyset$, and Π_2,Σ_2 are respectively Π'',Σ'' or $\emptyset$.

4.3 Strutural cuts

A structural cut is a generalised cut which is not a logical cut. The elimination of a structural cut R is obtained by reversing the order of the rules: one applies the rule R directly to the axioms containing occurences of the main formula of R. This elimination can be defined as a composition of proofs (suppose for example that the main formula is to the right):

$$\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Gamma \vdash A^y, \Delta\end{array} \qquad \begin{array}{c}\ldots.\\ \\ \\ \end{array}}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\, R$$

reduces to

$$\begin{array}{c}d_1[[A^w\vdash]e/\vdash A^y]\\ \vdots\\ \Gamma, \Gamma'' \vdash \Delta, \Delta''\end{array}$$

with $e := \dfrac{A^w \vdash A^y \qquad \ldots.}{\Gamma', A^w \vdash \Delta'}$, w is a new index and Γ'',Δ'' are respectively Γ',Δ' or $\emptyset$.

Remark. The procedure of elimination of structural cuts allows to transform a proof in free deduction into a proof which is "essentially" in sequent calculus: one has only to write the cuts as aplications of the cut–rule of sequent calculus.

4.4 Confluence and strong normalisation.

We have seen in 4.2 that there is a computational non–determinism in classical logic, which corresponds to the freedom for the choice of the inputs. From now on we will fix the inputs. For this we consider signed formulas A_l or A_r, where l and r are signs which indicate where the input is (the sign is considered as a part of the formula), and we suppose that the elimination of logical cuts is done according to the signs; for example

$$\dfrac{\dfrac{A_r\wedge B \vdash A_r\wedge B \qquad \begin{array}{c}d_1\\ \vdots\\ \Pi \vdash A_r{}^y, \Sigma\end{array} \qquad \begin{array}{c}d_2\\ \vdots\\ \Pi' \vdash B, \Sigma'\end{array}}{\Pi, \Pi' \vdash A_r\wedge B, \Sigma, \Sigma'}\,\wedge l \qquad \begin{array}{c}d_3\\ \vdots\\ \Pi'', A_r{}^x \vdash \Sigma''\end{array}}{\Pi, \Pi', \Pi'' \vdash \Sigma, \Sigma', \Sigma''}\,\wedge r$$

reduces to

$$\begin{array}{c}d_1[[A_r{}^x\vdash]d_3/\vdash A_r{}^y]\\ \vdots\\ \Pi, \Pi_2 \vdash \Sigma, \Sigma_2\end{array}$$

where Π_2,Σ_2 are respectively Π'',Σ'' or $\emptyset$.

Remark The semantic of formulas is not changed by the addition of signs: signs are just external annotations for cut–elimination without any effect on the proofs themselves.

Once the inputs are fixed, cut–elimination in free deduction becomes a deterministic computation mechanism for classical logic which is very like the one provided by natural deduction for intuitionistic logic (where the inputs are fixed by the syntax). In particular it enjoys the confluence and strong normalisation properties. Moreover these properties hold for the structural

and logical cuts separately. Here is a summary of the results proved in the next section.

Theorem A. The procedure of elimination of structural cuts enjoys:
– the confluence property, up to the order of logical cuts, and
– the strong normalisation property.

Theorem B. The procedure of elimination of logical cuts enjoys:
– the confluence property, and
– the strong normalisation property.

Theorem C. The procedure of elimination of generalised cuts enjoys:
– the confluence property, and
– the strong normalisation property.

Remark. These results apply in particular to proofs of sequent calculus and allow to compute corresponding cut–free proofs in sequent calculus, but the computation cannot be done inside sequent calculus! (i.e. in general, intermediate proofs are not proofs of sequent calculus). In fact, having free deduction in mind, we can define inside sequent calculus cut–elimination procedures giving the same results.

Remark. For the "unified" rules of FD' the situation is a little bit more complex. The choice of the inputs doesn't completely determine the situation in one case: if the secundary active formulas of a unified rule are both outputs, then the order between the two corresponding inputs is not determined by the signs; here is the case of the conjunction:

$$\dfrac{\dfrac{A_r \wedge B_r \vdash A_r \wedge B_r \qquad \overset{d_1}{\Pi \vdash A_r^{\,y},\ \Sigma} \qquad \overset{d_2}{\Pi' \vdash B_r^{\,z},\ \Sigma'}}{\Pi, \Pi' \vdash A_r \wedge B,\ \Sigma,\ \Sigma'}\ \wedge l \qquad \overset{d_3}{\Pi'',\ A_r^{\,x},\ B_r^{\,t} \vdash \Sigma''}}{\Pi,\ \Pi',\ \Pi'' \vdash \Sigma,\ \Sigma',\ \Sigma''}\ \wedge r$$

In this case we have two possible reductions which correspond to two ways of simulating the unified rule of FD' by the corresponding rules of FD (in this respect, unified rules seem not primitive).
There are many possible determinizations of the calculus: binary connectives bearing a priority between their components, or restrictions on the notion of proof.

5. PROOFS OF CONFLUENCE AND STRONG NORMALISATION.

Because of lack of place we cannot give full proofs of the results. We will give relatively detailed proofs for local confluence – which is the heart of the problem – and only sketch the rest. Confluence is deduced from local confluence and finiteness of developments (see [1]). We have choosen combinatorial proofs which seem to me more explicative than abstract ones.

In this section proofs are represented linearly as terms (as in typed lambda–calculus). A term is either an axiom or of the form $([X_1]d_1\ [X_2]d_2\ [X_3]d_3)$ or $([X_1]d_1\ [X_2]d_2)$ where d_i are terms, X_i are of the form $\vdash A$ or $A\vdash$, and $[]$ is a binding; this notation allows to represent the application of a rule to previous proofs d_i with an explicit mention of the active formulas (considered as bounded) together with their position in the sequent; for example,

$$\frac{\begin{array}{ccc} d_1 & d_2 & d_3 \\ \vdots & \vdots & \vdots \\ \Gamma,\ A\wedge B \vdash \Delta & \Pi \vdash A,\ \Sigma & \Pi' \vdash B,\ \Sigma' \end{array}}{\Gamma,\ \Pi,\ \Pi' \vdash \Delta,\ \Sigma,\ \Sigma'}$$

is linearly represented as

$$([A\wedge B\vdash]d_1\ [\vdash A]d_2\ [\vdash B]d_3).$$

For convenience, we often consider the intermediate expression $[X]d$ as a term. A proof ending with a (structural, logical, generalised) cut is called a (structural, logical, generalised) redex, and cut–elimination becomes reduction of redexes.

Convention: to simplify exposition we will often consider only the case of binary rules and state the properties up to obvious symetries.

5.1 Substitution.

Definition. For A^y not bounded in d, $d[[A^x\vdash]e/\vdash A^y]$ is defined inductively as follows

$\{A^z \vdash A^y\}[[A^x\vdash]e/\vdash A^y] = e[A^z/A^x]$

$\{C^u \vdash C^v\}[[A^x\vdash]e/\vdash A^y] = C^u \vdash C^v$ if $C^v \neq A^y$ (i.e. $C \neq A$ or $v \neq y$)

$\{[X]d\}[[A^x\vdash]e/\vdash A^y] = [X]d[[A^x\vdash]e/\vdash A^y]$

$(u\ v)[[A^x\vdash]e/\vdash A^y] = (u[[A^x\vdash]e/\vdash A^y]\ v[[A^x\vdash]e/\vdash A^y])$

Lemma 1 If $A^u \neq A^z$, then $e[[A^z\vdash]f/\vdash A^x][A^u/A^y] = e[A^u/A^y][[A^z\vdash]\{f[A^u/A^y]\}/\vdash A^x]$; in particular, if A^y is not free in f, then $e[[A^z\vdash]f/\vdash A^x][A^u/A^y] = e[A^u/A^y][[A^z\vdash]f/\vdash A^x]$

Lemma 2

(i) If A^y and A^x are not free in f and $A^y \neq B^z$, then
$d[[A^x\vdash]e/\vdash A^y][[B^t\vdash]f/\vdash B^z] = d[[B^t\vdash]f/\vdash B^z][[A^x\vdash]\{e[[B^t\vdash]f/\vdash B^z]\}/\vdash A^y]$.

(ii) If A^y and A^x are not free in f and A^y, B^z are not related by an axiom in d, then
$d[[A^x\vdash]e/\vdash A^y][[\vdash B^t]f/B^z\vdash] = d[[\vdash B^t]f/B^z\vdash][[A^x\vdash]\{e[[\vdash B^t]f/B^z\vdash]\}/\vdash A^y]$.

5.2 Structural reduction.

We will have to consider proofs up to the order of the logical cuts; this equivalence relation is denoted by $\sim$. If r is redex $([A^x\vdash]d_1\ d_2)$ or $([\vdash A^x]d_1\ d_2)$, then $[A^x\vdash]d_1$ is called the function of r, d_2 the argument of r and A^x the binder of r.

Definition. The contractum $d_1\langle d_2/A^x\vdash\rangle$ of a structural redex $([A^x\vdash]d_1\ d_2)$ is $d_1[\beta/A^x\vdash]$ with $\beta = [\vdash A^u]([A^v\vdash]\{A^v\vdash A^u\}\ d_2)$ for A^u, A^v not occuring in d_2. Structural reduction $\overset{s}{\triangleright}$ is the reflexive and transitive closure of the one–step structural reduction $\overset{s}{\triangleright}_1$ defined by:

$([A^x\vdash]d_1\ d_2) \overset{s}{\triangleright}_1 d_1\langle d_2/A^x\vdash\rangle$, if $([A^x\vdash]d_1\ d_2)$ is a strutural redex;

$(d_1\ d_2) \overset{s}{\triangleright}_1 (d_1\ d_2')$, if $d_2 \overset{s}{\triangleright}_1 d_2'$;

$(d_1\ d_2) \overset{s}{\triangleright}_1 (d_1'\ d_2)$, if $d_1 \overset{s}{\triangleright}_1 d_1'$;

$[X]d \overset{s}{\triangleright}_1 [X]d'$, if $d \overset{s}{\triangleright}_1 d'$.

Remark 3 The contractum is defined for structural redexes, but more generally, if $u = ([A^x\vdash]d_1\ d_2)$ and $v = d_1\langle d_2/A^x\vdash\rangle$, then $u \overset{s}{\triangleright} v$ or $v \sim u$; in fact one of the following situations hold:

– u is a structural redex and $u \overset{s}{\triangleright}_1 v$;

– u is a logical redex and $u \sim v$;

– u is not a redex and $u = v$.

Lemma 4 If $u \overset{s}{\triangleright}_1 v$ then $u[A^x/A^y] \overset{s}{\triangleright}_1 v[A^x/A^y]$.

Lemma 5

(i) If $u \overset{s}{\triangleright}_1 v$ then there exist u' and v' such that $u' \sim v'$, $u[[A^y\vdash]d/\vdash A^x] \overset{s}{\triangleright} u'$ and $v[[A^y\vdash]d/\vdash A^x] \overset{s}{\triangleright} v'$ (in the case where the binder of the reduced redex of u is not related to A^x in u, we have in fact $u[[A^y\vdash]d/\vdash A^x] \overset{s}{\triangleright}_1 v[[A^y\vdash]d/\vdash A^x]$).

(ii) If $u \overset{s}{\triangleright}_1 v$, then there exists u' such that $u\langle e/\vdash A^x\rangle \overset{s}{\triangleright}_1 u'$ and $v\langle e/\vdash A^x\rangle \sim u'$.

Proof. (i) We prove the result by induction on $\overset{s}{\triangleright}_1$. Let $u_1 = u[[A^y\vdash]d/\vdash A^x]$ and $v_1 = v[[A^y\vdash]d/\vdash A^x]$. The only interesting case is when v is the contractum of u.

1. $u = ([\vdash B^z]d_1\ d_2)$ and $B^z \neq A^x$.

We have

$v = d_1[\beta/\vdash B^z]$ with $\beta = [B^u\vdash]([\vdash B^v]\{B^u\vdash B^v\}\ d_2)$

$u_1 = ([\vdash B^z]\{d_1[[A^y\vdash]d/\vdash A^x]\}\ d_2[[A^y\vdash]d/\vdash A^x])$;

$v_1 = d_1[\beta/\vdash B^z][[A^y\vdash]d/\vdash A^x]$;

and u_1 reduces to u' where

$u' = d_1[[A^y\vdash]d/\vdash A^x][[B^u\vdash]([\vdash B^v]\{B^u\vdash B^v\}\ d_2[[A^y\vdash]d/\vdash A^x])/\vdash B^z]$

$= d_1[[A^y\vdash]d/\vdash A^x][\beta[[A^y\vdash]d/\vdash A^x]/\vdash B^z]$.

Because B^z is not free in d and $B^z \neq A^x$, we have $u' = v_1$ by lemma 2.(i).

2. $u = ([B^z\vdash]d_1\ d_2)$ with $B \neq A$ (and more generally with B^z not related to A^x in d_1).

We have

$v = d_1[\beta/B^z\vdash]$ with $\beta = [\vdash B^u]([B^v\vdash]\{B^v\vdash B^u\}\ d_2)$

$u_1 = ([B^z\vdash]\{d_1[[A^y\vdash]d/\vdash A^x]\}\ d_2[[A^y\vdash]d/\vdash A^x])$;

$v_1 = d_1[\beta/B^z\vdash][[A^y\vdash]d/\vdash A^x]$;

and u_1 reduces to u' where

$u' = d_1[[A^y\vdash]d/\vdash A^x][[\vdash B^u]([B^v\vdash]\{B^v\vdash B^u\}\ d_2[[A^y\vdash]d/\vdash A^x])/B^z\vdash]$

$= d_1[[A^y\vdash]d/\vdash A^x][\beta[[A^y\vdash]d/\vdash A^x]/B^z\vdash]$.

Because B^z is not free in d and not related to A^x in d_1, we have $u' = v_1$ by lemma 2.(ii).

3. $u = ([A^z\vdash]d_1\ d_2)$.

We have

$v = d_1[\beta/A^z\vdash]$ with $\beta = [\vdash A^u]([A^v\vdash]\{A^v\vdash A^u\}\ d_2)$

$u_1 = ([A^z\vdash]\{d_1[[A^y\vdash]d/\vdash A^x]\}\ d_2[[A^y\vdash]d/\vdash A^x])$;

$v_1 = d_1[\beta/A^z\vdash][[A^y\vdash]d/\vdash A^x]$;

and u_1 reduces to u' where

$u' = d_1[[A^y\vdash]d/\vdash A^x][[\vdash A^u]([A^v\vdash]\{A^v\vdash A^u\}\ d_2[[A^y\vdash]d/\vdash A^x])/A^z\vdash]$

$= d_1[[A^y\vdash]d/\vdash A^x][\beta[[A^y\vdash]d/\vdash A^x]/A^z\vdash]$.

We prove by induction on d_1 that there exists v' such that $v_1 \overset{s}{\triangleright} v'$ and $v' \sim u'$. The only interesting case is when d_1 is an axiom. If d_1 is an axiom other than $A^z \vdash A^x$, then A^z and A^x are not related in d_1 and we have in fact $u' = v_1$ by lemma 2.(ii)(of course A^z is not free in d). Suppose that d_1 is the axiom $A^z \vdash A^x$. In this case we have

$v_1 = ([A^v\vdash]\{A^v\vdash A^x\}\ d_2)[[A^y\vdash]d/\vdash A^x]$

$= ([A^v\vdash]d[A^v/A^y]\ d_2[[A^y\vdash]d/\vdash A^x])$

$u' = d[A^z/A^y][\beta[[A^y\vdash]d/\vdash A^x]/A^z\vdash]$

$= d[A^z/A^y][[\vdash A^u]([A^v\vdash]\{A^v\vdash A^u\}\ d_2[[A^y\vdash]d/\vdash A^x])/A^z\vdash]$.

Let $w = d[A^v/A^y][\gamma/A^v\vdash]$, where $\gamma = [\vdash A^u]([A^w\vdash]\{A^w\vdash A^u\}\ d_2[[A^y\vdash]d/\vdash A^x])$. By remark 3, we have $v_1 \overset{s}{\triangleright} w$ or $v_1 \sim w$; because A^v and A^z are not free in d, we also have $u' = w$; therefore we have $v_1 \overset{s}{\triangleright} u'$ or $v_1 \sim u'$: in the first case we take $v' = u'$ and in the second $v' = v_1$.

(ii) We just have to check that in case 3 of the previous proof we have $u' \sim v_1$ if d is $([\vdash A^w]\{A^y \vdash A^w\}\ e)$. The only interesting case is when d_1 is the axiom $A^z \vdash A^x$. In this case we have

$v_1 = ([A^v\vdash]\{A^v\vdash A^x\}\ d_2)[[A^y\vdash]d/\vdash A^x]$

$= ([A^v\vdash]([\vdash A^w]\{A^v \vdash A^w\}\ e)\ d_2[[A^y\vdash]d/\vdash A^x])$

$u' = ([\vdash A^w]\{A^z \vdash A^w\}\ e)[\beta[[A^y\vdash]d/\vdash A^x]/A^z\vdash]$

$= ([\vdash A^w]\{A^z \vdash A^w\}\ e)[[\vdash A^u]([A^v\vdash]\{A^v\vdash A^u\}\ d_2[[A^y\vdash]d/\vdash A^x])/A^z\vdash]$

$= ([\vdash A^w]([A^v\vdash]\{A^v\vdash A^w\}\ d_2[[A^y\vdash]d/\vdash A^x])\ e)$.

Therefore $v_1 \sim u'$.

<u>Lemma 6</u> If $u \overset{s}{\triangleright}_1 v$ then $w[[A^y\vdash]u/\vdash A^x] \overset{s}{\triangleright} w[[A^y\vdash]v/\vdash A^x]$.

<u>Proof</u>. The result is proved by induction on w. The only interesting case is when w is an axiom. Let $w_1 = w[[A^y\vdash]u/\vdash A^x]$ and $w_2 = w[[A^y\vdash]v/\vdash A^x]$. If $w = C^u \vdash C^v$ with $C^v \neq A^x$, then $w_1 = w_2 = w$. If $w = A^z \vdash A^x$, then we have $w_1 = u[A^z/A^y]$ and $w_2 = v[A^z/A^y]$, and therefore $w_1 \overset{s}{\triangleright} w_2$ by lemma 4.

<u>Lemma 7</u> If $u \overset{s}{\triangleright}_1 v$ and $u \sim u'$, then there exists v' such that $v \sim v'$ and $u' \overset{s}{\triangleright}_1 v'$.

<u>Theorem 1</u> Structural reduction is locally confluent, up to the order of the logical redexes, i.e. if $u \overset{s}{\triangleright}_1 u_1$ and $u \overset{s}{\triangleright}_1 u_2$, then there exist v_1 and v_2 such that $v_1 \sim v_2$, $u_1 \overset{s}{\triangleright} v_1$ and $u_2 \overset{s}{\triangleright} v_2$.

<u>Proof</u>. The proof is by induction on u. The only interesting case is when u is one of the redexes which are reduced, say $u = ([A^x\vdash]d_1\ d_2)$ and $u_1 = d_1\langle d_2/A^x\vdash\rangle$. We consider the different possibilities for u_2.

1. $u_2 = (\llbracket A^x \vdash \rrbracket d_1\ d_2')$ with $d_2 \overset{s}{\triangleright}_1 d_2'$.
In this case u_2 is a structural redex and $u_2 \overset{s}{\triangleright}_1 v = d_1\langle d_2'/A^x\vdash\rangle$; by lemma 6, we have also $u_1 \overset{s}{\triangleright} v$.
2. $u_2 = (\llbracket A^x \vdash \rrbracket d_1'\ d_2)$ with $d_1 \overset{s}{\triangleright}_1 d_1'$.
Let $u_2' = d_1'\langle d_2/A^x\vdash\rangle$. Because $d_1 \overset{s}{\triangleright}_1 d_1'$, there exist (by lemma 5) v_1 such that $u_1 \overset{s}{\triangleright} v_1$ and $v_1 \sim u_2'$. By remark 3, we also have $u_2 \overset{s}{\triangleright} u_2'$ or $u_2 \sim u_2'$: in the first case we take $v_2 = u_2'$; in the second case we take $v_2 = u_2$; in each case we have $v_1 \sim v_2$, $u_1 \overset{s}{\triangleright} v_1$ and $u_2 \overset{s}{\triangleright} v_2$.

Theorem 2 Structural reduction enjoys the strong normalisation property.

Proof (sketch of). We prove that each structural reduction sequence of a term t is finite, by induction on the number of structural redexes of t. Suppose that t has at least one redex (otherwise the result is trivial). Consider the redex $r = (\llbracket X \rrbracket u_1\ u_2)$ or $r = (\llbracket X \rrbracket u_1\ u_2\ u_3)$ whose first argument u_2 is the rightmost one. We label t as follows: the binder of r receives the label 1, and the others a label $\neq 1$. Let t' obtained from t by reducing r; t' has less structural redexes than t and therefore by induction hypothesis, each structural reduction sequence of t' is finite.
Suppose now that t has an infinite structural reduction sequence (t_i). We consider a function φ between terms which reduces every structural redex whose binder is labelled 1. Using the fact the sequences of reduction of structural redexes whose binder is labelled 1 are finite, we can extract an infinite subsequence (s_j) of (t_i) such that $s_0 = t$ and for each j, there exists v_j such that $\varphi s_j \overset{s}{\triangleright}_1 v_j$ and $v_j \sim \varphi s_{j+1}$. Using lemma 7, we can transform (s_j) into an infinite structural reduction sequence of t'; this is impossible and therefore each structural reduction sequence of t is finite.

Corollary 3 Structural reduction is confluent up to the order of the logical redexes, i.e. if $u \overset{s}{\triangleright} u_1$ and $u \overset{s}{\triangleright} u_2$, then there exist v_1 and v_2 such that $v_1 \sim v_2$, $u_1 \overset{s}{\triangleright} v_1$ and $u_2 \overset{s}{\triangleright} v_2$.

5.3 Logical reduction.

One defines notions of function, argument and binder for logical redexes. For example, if r is the redex $(\llbracket \vdash C_r \wedge D \rrbracket(\llbracket C_r \wedge D \vdash \rrbracket\{C_r \wedge D \vdash C_r \wedge D\}\ \llbracket \vdash C_r^u \rrbracket d_1\ \llbracket \vdash D \rrbracket d_2)\ \llbracket C_r^v \vdash \rrbracket d_3)$, then $\llbracket \vdash C_r^u \rrbracket d_1$ is called the function of r, $\llbracket C_r^v \vdash \rrbracket d_3$ the argument and C_r^u the binder; the contractum of r is $d_1[\llbracket C_r^v \vdash \rrbracket d_3/\vdash C_r^u]$.

Definition. Logical reduction $\overset{l}{\triangleright}$ is the reflexive and transitive closure of the one-step logical reduction $\overset{l}{\triangleright}_1$ defined by:
$d \overset{l}{\triangleright}_1 d'$, if d a logical redex and d' its contractum;
$(d_1\ d_2) \overset{l}{\triangleright}_1 (d_1\ d_2')$, if $d_2 \overset{l}{\triangleright}_1 d_2'$;
$(d_1\ d_2) \overset{l}{\triangleright}_1 (d_1'\ d_2)$, if $d_1 \overset{l}{\triangleright}_1 d_1'$;
$\llbracket X \rrbracket d \overset{l}{\triangleright}_1 \llbracket X \rrbracket d'$, if $d \overset{l}{\triangleright}_1 d'$.

Lemma 8 If $u \overset{l}{\triangleright}_1 v$ then $u[A^x/A^y] \overset{l}{\triangleright}_1 v[A^x/A^y]$.

Lemma 9

If $u \overset{1}{\triangleright_1} v$ then $u[[A_r^y\vdash]d/\vdash A_r^x] \overset{1}{\triangleright_1} v[[A_r^y\vdash]d/\vdash A_r^x]$ and $u[[\vdash A_1^y]d/A_1^x\vdash] \overset{1}{\triangleright_1} v[[\vdash A_1^y]d/A_1^x\vdash]$.

Proof. We prove the result by induction on $\overset{1}{\triangleright_1}$ for a right input (the proof is the same for a left one). Let $u_1 = u[[A_r^y\vdash]d/\vdash A_r^x]$ and $v_1 = v[[A_r^y\vdash]d/\vdash A_r^x]$. The only interesting case is when u is the logical redex which is reduced; suppose for example it is a redex for a conjunction.

1. $u = ([\vdash C_r\wedge D]([C_r\wedge D\vdash]\{C_r\wedge D \vdash C_r\wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2)\ [C_r^v\vdash]d_3)$.

We have

$v = d_1[[C_r^v\vdash]d_3/\vdash C_r^u]$;

$u_1 = ([\vdash C_r\wedge D]([C_r\wedge D\vdash]\{C_r\wedge D \vdash C_r\wedge D\}\ [\vdash C_r^u]d_1'\ [\vdash D]d_2')\ [C_r^v\vdash]d_3')$

with $d_i' = d_i[[A_r^y\vdash]d/\vdash A_r^x]$ for i=1,2,3;

$v_1 = d_1[[C_r^v\vdash]d_3/\vdash C_r^u][[A_r^y\vdash]d/\vdash A_r^x]$;

and $u_1 \overset{1}{\triangleright_1} w$ where

$w = d_1'[[C_r^v\vdash]d_3'/\vdash C_r^u]$

$= d_1[[A_r^y\vdash]d/\vdash A_r^x][[C_r^v\vdash]\{d_3[[A_r^y\vdash]d/\vdash A_r^x]\}/\vdash C_r^u]$.

Because $C_r^u \neq A_r^x$ and C_r^u is not free in d, we have by lemma 2.(i) $w = v_1$ and therefore $u_1 \overset{1}{\triangleright_1} v_1$.

2. $u = ([\vdash C_1\wedge D]([C_1\wedge D\vdash]\{C_1\wedge D \vdash C_1\wedge D\}\ [\vdash C_1^u]d_1\ [\vdash D]d_2)\ [C_1^v\vdash]d_3)$.

We have

$v = d_3[[\vdash C_1^u]d_1/C_1^v\vdash]$;

$u_1 = ([\vdash C_1\wedge D]([C_1\wedge D\vdash]\{C_1\wedge D \vdash C_1\wedge D\}\ [\vdash C_1^u]d_1'\ [\vdash D]d_2')\ [C_1^v\vdash]d_3')$

with $d_i' = d_i[[A_r^y\vdash]d/\vdash A_r^x]$ for i=1,2,3;

$v_1 = d_3[[\vdash C_1^u]d_1/C_1^v\vdash][[A_r^y\vdash]d/\vdash A_r^x]$;

and $u_1 \overset{1}{\triangleright_1} w$ where

$w = d_3'[[\vdash C_1^u]d_1'/C_1^v\vdash]$

$= d_3[[A_r^y\vdash]d/\vdash A_r^x][[\vdash C_1^u]\{d_1[[A_r^y\vdash]d/\vdash A_r^x]\}/C_1^v\vdash]$.

By choice of the inputs $C_1 \neq A_r$ and therefore C_1^v is not related to A_r^x in d_3; moreover C_1^v is not free in d, and therefore we have by lemma 2.(ii) $w = v_1$ and $u_1 \overset{1}{\triangleright_1} v_1$.

Lemma 10 If $u \overset{1}{\triangleright_1} v$ then $w[[A^y\vdash]u/\vdash A^x] \overset{1}{\triangleright} w[[A^y\vdash]v/\vdash A^x]$.

Theorem 4 Logical reduction is locally confluent, i.e. if $u \overset{1}{\triangleright_1} u_1$ and $u \overset{1}{\triangleright_1} u_2$, then there exists v such that $u_1 \overset{1}{\triangleright} v$ and $u_2 \overset{1}{\triangleright} v$.

Proof. The proof is by induction on u. The only interesting case is when u is one of the redexes which are reduced. Suppose for instance that

$u = ([\vdash C_r\wedge D]([C_r\wedge D\vdash]\{C_r\wedge D \vdash C_r\wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2)\ [C_r^v\vdash]d_3)$;

$u_1 = d_1[[C_r^v\vdash]d_3/\vdash C_r^u]$.

We consider the different possibilities for u_2.

1. $u_2 = ([\vdash C_r\wedge D]([C_r\wedge D\vdash]\{C_r\wedge D \vdash C_r\wedge D\}\ [\vdash C_r^u]d_1'\ [\vdash D]d_2)\ [C_r^v\vdash]d_3)$ with $d_1 \overset{1}{\triangleright_1} d_1'$.

Let v be $d_1'[[C_r^v\vdash]d_3/\vdash C_r^u]$. We have $u_2 \overset{1}{\triangleright_1} v$ and by lemma 9, $u_1 \overset{1}{\triangleright_1} v$.

2. $u_2 = ([\vdash C_r\wedge D]([C_r\wedge D\vdash]\{C_r\wedge D \vdash C_r\wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2')\ [C_r^v\vdash]d_3)$ with $d_2 \overset{1}{\triangleright_1} d_2'$.

Let v be $d_1[[C_r^v\vdash]d_3/\vdash C_r^u]$. We have $u_2 \overset{1}{\triangleright_1} v$ and $u_1 = v$.

3. $u_2 = ([\vdash C_r \wedge D]([C_r \wedge D \vdash]\{C_r \wedge D \vdash C_r \wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2)\ [C_r^v \vdash]d_3')$ with $d_3 \overset{1}{\triangleright_1} d_3'$.
Let v be $d_1[[C_r^v \vdash]d_3'/\vdash C_r^u]$. We have $u_2 \overset{1}{\triangleright_1} v$ and by lemma 10, $u_1 \overset{1}{\triangleright} v$.

Theorem 5 Logical reduction enjoys the finiteness of developments property.

Proof (sketch of). Consider a term t labelled as follows: the binders of the logical redexes are labelled 1 or 2 and the other ones 0. We call i–redex (resp. $\bar{i}$–redex) a logical redex whose binder is labelled i (resp. $\neq$i). We prove that each reduction sequence of $\bar{0}$–redexes of t is finite, by induction on the number of $\bar{0}$–redexes of t. Suppose that t has at least one $\bar{0}$–redex (otherwise the result is trivial). Consider the $\bar{0}$–redex r whose argument is the rightmost one (we suppose for simplicity that the argument is always to the right of the function). We change the labels of the $\bar{0}$–redexes as follows: the binder of r receives the label 1, and the binders of the other logical redexes, the label 2. Let t' obtained from t by reducing r; t' has less $\bar{0}$–redexes than t and therefore by induction hypothesis, each reduction sequence of $\bar{0}$–redexes of t' is finite.
Suppose now that t has an infinite sequence (t_i) of reduction of $\bar{0}$–redexes. We get a contradiction in the same way as for theorem 2 using a function ψ between terms which reduces every 1–redex

Corollary 6 Logical reduction is confluent, i.e. if $u \overset{1}{\triangleright} u_1$ and $u \overset{1}{\triangleright} u_2$, then there exist v such $u_1 \overset{1}{\triangleright} v$ and $u_2 \overset{1}{\triangleright} v$.

5.4 Generalised reduction.

Definition. Generalised reduction $\triangleright$ is the reflexive and transitive closure of the one–step generalised reduction $\triangleright_1$ defined by: $u \triangleright_1 v$ if and only if $u \overset{s}{\triangleright_1} v$ or $u \overset{1}{\triangleright_1} v$.

Lemma 11 If $u \overset{1}{\triangleright_1} v$, then there exists v' such that $u\langle d/\vdash A^x\rangle \triangleright v'$ and $v' \sim \overset{1}{\triangleright_1} v\langle d/\vdash A^x\rangle$.

Theorem 7 Generalised reduction is locally confluent, up to the order of the logical redexes, i.e. if $u \triangleright_1 u_1$ and $u \triangleright_1 u_2$, then there exist v_1 and v_2 such that $v_1 \sim v_2$, $u_1 \triangleright v_1$ and $u_2 \triangleright v_2$.

Proof. The proof is by induction on u. The only interesting case is when u_1 (or u_2) is the contractum of u. If the reduced redexes are both structural ones or both logical ones we apply the theorems 1 and 4. We consider the remaining cases.
1. $u \overset{1}{\triangleright_1} u_1$ and $u \overset{s}{\triangleright_1} u_2$.
Suppose for instance that

$u = ([\vdash C_r \wedge D]([C_r \wedge D \vdash]\{C_r \wedge D \vdash C_r \wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2)\ [C_r^v \vdash]d_3)$;

$u_1 = d_1[[C_r^v \vdash]d_3/\vdash C_r^u]$.

We consider the different possibilities for u_2.
1.1. $u_2 = ([\vdash C_r \wedge D]([C_r \wedge D \vdash]\{C_r \wedge D \vdash C_r \wedge D\}\ [\vdash C_r^u]d_1'\ [\vdash D]d_2)\ [C_r^v \vdash]d_3)$ with $d_1 \overset{s}{\triangleright_1} d_1'$.
Let v be $d_1'[[C_r^v \vdash]d_3/\vdash C_r^u]$. Because $d_1 \overset{s}{\triangleright_1} d_1'$, there exist (by lemma 5) v_1 and v_2 such that $v_1 \sim v_2$,

$u_1 \overset{s}{\triangleright} v_1$ and $v \overset{s}{\triangleright} v_2$; because $u_2 \overset{1}{\triangleright}_1 v$, we have in fact $u_1 \triangleright v_1$ and $u_2 \triangleright v_2$.

1.2. $u_2 = ([\vdash C_r \wedge D]([C_r \wedge D \vdash]\{C_r \wedge D \vdash C_r \wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2')\ [C_r^v \vdash]d_3)$ with $d_2 \overset{s}{\triangleright}_1 d_2'$.

Let v be $d_1[[C_r^v \vdash]d_3/\vdash C_r^u]$. We have $u_2 \overset{1}{\triangleright}_1 v$ and $u_1 = v$.

1.3. $u_2 = ([\vdash C_r \wedge D]([C_r \wedge D \vdash]\{C_r \wedge D \vdash C_r \wedge D\}\ [\vdash C_r^u]d_1\ [\vdash D]d_2)\ [C_r^v \vdash]d_3')$ with $d_3 \overset{s}{\triangleright}_1 d_3'$.

Let v be $d_1[[C_r^v \vdash]d_3'/\vdash C_r^u]$. We have $u_2 \overset{1}{\triangleright}_1 v$ and by lemma 6, $u_1 \overset{s}{\triangleright} v$.

2. $u \overset{s}{\triangleright}_1 u_1$ and $u \overset{1}{\triangleright}_1 u_2$.

Suppose that $u = ([A^x \vdash]d_1\ d_2)$ and $u_1 = d_1\langle d_2/A^x \vdash\rangle$. We consider the different possibilities for u_2.

2.1. $u_2 = ([A^x \vdash]d_1\ d_2')$ with $d_2 \overset{1}{\triangleright}_1 d_2'$.

In this case u_2 is a structural redex and $u_2 \overset{s}{\triangleright}_1 v = d_1\langle d_2'/A^x \vdash\rangle$; by lemma 10, we have also $u_1 \overset{1}{\triangleright} v$.

2.2. $u_2 = ([A^x \vdash]d_1'\ d_2)$ with $d_1 \overset{1}{\triangleright}_1 d_1'$.

Let $v = d_1'\langle d_2/A^x \vdash\rangle$. By lemma 11, there exists v_1 such that $u_1 \triangleright v_1$ and $v_1 \sim v$. By remark 3, we also have $u_2 \overset{s}{\triangleright} v$ or $u_2 \sim v$: in the first case we take $v_2 = v$ and in the second one $v_2 = u_2$; in each case we have $v_1 \sim v_2$, $u_1 \triangleright v_1$ and $u_2 \triangleright v_2$.

Theorem 8 Generalised reduction enjoys the finiteness of developments property.

Proof. The proof is essentially a combination of the proofs of theorem 2 and 5.

Corollary 9 Generalised reduction is confluent, i.e. if $u \triangleright u_1$ and $u \triangleright u_2$, then there exists v such that $u_1 \triangleright v$ and $u_2 \triangleright v$.

Theorem 10 Generalised reduction enjoys strong normalisation property.

Proof (sketch of). We call strict a term in which every binder has at least one occurence. The first step is the reduction of the problem to strict terms, by interpreting arbitrary terms into strict terms in a way which preserves the reduction sequences. In a second step we prove the strong normalisation for strict terms in the following combinatorial way. We prove that each reduction sequence of a strict term t is finite, by induction on $\Phi(t) = (d(t),n(t))$ with lexicographic ordering, where d(t) is the maximum degree of the redexes of t (the degree of a redex is the length of the main formula of the corresponding cut) and n(t) is the number of redexes of degree d(t). Let r be a redex of maximal degree whose first argument is the rightmost one. We label t as follows: the binder of r receives label 1, and the other binders, a label $\neq 1$. Let t' obtained from t by reducing r; we have $\Phi(t') < \Phi(t)$ and by induction hypothesis, each reduction sequence of t' is finite.

Suppose now that t has an infinite reduction sequence (t_i). We get a contradiction in the same way as for theorem 2 using a function Θ between terms which reduces every 1–redex (in order to "transform" (t_i) into an infinite reduction sequence of t', we use the fact that the terms are strict and the finiteness of developments).

RELATED WORKS / FURTHER WORKS

There is a growing interest in the extraction of programs from classical proofs in particular through the works of T. Griffin and C. Murthy. Though not directly related, our work has been

partly motivated by this idea which is in the air at the moment.

In a parallel non communicating work, J.Y. Girard has built a new computational system for classical logic LC, based on an operational semantic. It could be interesting to see whether FD and LC share some properties.

Among expected applications of FD are: the extraction of programs from classical proofs written in FD; the study of proof search strategies from the viewpoint of FD; the use of FD as a metasystem for studying usual logical systems.

BIBLIOGRAPHY.

[1] BARENDREGT, The lambda calculus, North–Holland, 1985.

[2] GIRARD, Proof theory and logical complexity, Bibliopolis, 1987.

[3] HOWARD, The formulae–as–types notion of construction, in "To HB Curry...", Academic Press, 1980.

[4] PRAWITZ, Natural deduction, Almqvist&Wiksell, 1965.