Contemporary Mathematics
Volume 92, 1989

TOWARDS A GEOMETRY OF INTERACTION

Jean-Yves Girard

dedicated to dag prawitz

Abstract : the paper presents a program for proof-theory, inspired by its growing connections with computer science ; what follows can therefore be seen as an updating of the -now classical- paradigms of Hilbert and Brouwer. Such an illustrious company is a bit daunting, and requires at least some evidence for the program : such evidence is based on the author's recent work on ***linear logic*** *[7] and on further developments, e.g. [8], [9]. In two words, we start from the idea that both Hilbert and Brouwer have still something to say (and not as antagonistic things as one could expect) provided one reacts against the* ***reductionism*** *of the former, and the* ***subjectivism*** *of the latter. The program is essentially about the development of a logic of* ***actions****, i.e. of non-reusable facts (versus* ***situations****) : we will accept the intuitionistic dogma that the meaning of a formula is in a proof of it (and not in its truth). The intuitionistic tradition understands proofs as subjectivistic entities, and develops an ideology of* ***intensionality****, which is often nothing more than an alibi for* ***taxonomy****, whereas one may reasonably advocate that proofs are the written trace of underlying geometrical structures. From formalism we shall keep the dogma of* ***finitism****, which was pushed to absurdity by Hilbert because he aimed at an absolute elimination of infinity : we now know that his proposed task, as carried out by Gentzen, involves a finite dynamics whose eventual behaviour is so unpredictible that only the reintroduction of infinite tools -more or less equivalent to the one under elimination- can master it. What must be kept of finitism is the idea of replacement of static infinite situations by finite dynamic actions ; this finite dynamics should lie in the geometrical structure of Gentzen's* ***Hauptsatz*** *which precisely eliminates the use of infinity in proofs.*

I. basic logical commitments

Before discussing the logical problems that will lead us to a drastic reformulation of logic, let us explain why such essential points have been overlooked by the whole logic tradition (including the author himself, who first found linear logic as a technical decomposition of intuitionistic logic, and only later on reconstructed a kind of commonplace justification for it). The reason has to be searched for in the obsession of *Grundlagen*, i.e. the furious reductionism under Hilbert's flag : since it was possible to *reduce* the formal core of any scientific activity to mathematics, it has been assumed that it was enough to analyze mathematics. Surely -in the spirit the Jivaro ideology- a reduction of mathematics would have induced a reduction of the formal core of other sciences. But reductionism in mathematics failed, and the reduction of -say the formal core of physics- to mathematics is simply a lemma in view of a wrong theorem. In fact, this reduction was a very awkward one, not taking care of the fact that the meaning of implication in real life or physical sciences has nothing to do with its familiar mathematical meaning : it is only through very heavy and *ad hoc* paraphrases that real implication may be put into mathematics (usually the paraphrase is done by adding a parameter for an extraneous time). The logical laws extracted from mathematics are only adapted to eternal truths ; the same principles applied in real life, easily lead to absurdity, because of the interactive (causal) nature of real implication.

I.1. platonism

One usually calls platonism the naive ideology shared by mathematicians about their field :

there is an external (ideal) world formulas are about.

The name "platonism" is a bit unfair to Plato who was not so simple-minded. Anyway, with this external stable infinite world, statements are true or false (independently of our ability to check them), and this justifies principles like

tertium non datur : $A \vee \neg A$

which is the core of *classical logic*, of universal use. Even if there is a lot of criticism to address to this logic, it is still our ultimate mathematical reference : even the most rabid constructivist must acknowledge that. The main problem with platonism is that it leaves no room for the very heart of mathematics, namely *proofs*. If "reality" is prior to anything else, proofs should be seen as a subjective process of understanding the real world. This is not a very satisfactory status, and we are lead to seek a less naive ontology for mathematics. The price to pay for that will be a farewell to the principles of classical logic. Although it is not excluded that classical logic could be compatible with the more elaborated viewpoints to be discussed below, there seems to be serious obstacles to its "constructivization" (however, the fact that linear logic is symmetric w.r.t. linear negation makes the situation a bit less hopeless).

I.2. intuitionism

The intuitionist paradigm is to dump the external world to focus on proofs. The fact that "reality" -which was anyway an abstraction- no longer finds a satisfactory status, has been interpreted in a subjectivistic way (leading to unbelievable nonsense, e.g. Brouwer's "creative subject"). But this relation of intuitionism with some kind of spiritualism is merely a historical accident : remember that at the beginning of the century, there was a lot of metaphysics about the actual nature of the world, matter or energy ; this was a hidden way to be for or against religion and the same kind of opposition existed among logicians between scientism (Hilbert) and mysticism (Brouwer). Brouwer's excessive ideological commitments should be interpreted as a defensive attitude against the spirit of the times -the Absolute Triumph of Science-. Just to say that subjectivism is far from being the only possible reading of Brouwer. One of the greatest ideas in logic (which has not yet received the mathematical treatment it deserves) is Heyting's *semantics of proofs*, which can be summarized by the slogan

proofs as functions.

Typically a proof of $A \Rightarrow B$ is a function mapping proofs of A into proofs of B. This explanation has been thought of as incomplete, since there is no way of deciding whether or not a given function maps proofs of A to proofs of B. But let us observe that :

a definite answer to the problem of "proving that a proof is a proof" would induce a *reduction* of intuitionism to a fixed formal system, which is absurd ;

the criticism to Heyting's paradigm relies on a subjectivistic attitude : I must recognize a proof when I see one of them. But, as we already mentioned, subjectivism is not the only issue : if a proof of A is a program enjoying specification A, we must accept the fact that, in most cases, a program will meet a given specification, but that there will be no way of checking it.

There is a more serious objection to Heyting's paradigm, namely the word "function". The standard acception of this term is "functional graph", while obviously Heyting meant something else. One usually speaks of *intensionality* with a clear subjective background : the function is given together with a description, and this widely opens the door for any kind of *taxonomy*. At that point, Heyting's semantics becomes a bubble-gum, since taxonomy allows us to distinguish between "a" and "A", 1 + 2 and 2 + 1, or even between "A" and "A", since they occur at different places.

I.3. actions

Since the paradigm "function & description" is clearly deficient, we shall propose another one :

proofs as actions.

The term action has to be understood with its familiar meaning (and not with a specific technical one like "group action"). Our program is essentially to try to give a precise mathematical contents to this expression. In the sequel, we shall try to give some hints as to the solution of this problem. The main difficulties will be :

to make a clear distinction between functions and actions

to try to get rid of taxonomy in the description of actions, i.e. to find out what is "beyond" syntax.

The logical twist from functions to actions leads us to linear logic.

II. Linear logic : a logic of action

II.1. actions versus situations

Classical and intuitionistic logics deal with *stable* truths :

If A and $A \Rightarrow B$, then B, *but* A *still holds.*

This is perfect in mathematics, but wrong in real life, since real implication is *causal*. A causal implication cannot be iterated since the conditions are modified after its use ; this process of modification of the premises (conditions) is known in physics as *reaction*. For instance, if A is to spend $1 on a pack of cigarettes and B is to get them, you lose $1 in this process, and you cannot do it a second time. The reaction here was that $1 went out of your pocket. The first objection to that view is that there are in mathematics, in real life, cases where reaction does not exist or can be neglected. Such cases are *situations* in the sense of stable truths. Our logical refinements should not prevent us to cope with situations, and there will be a specific kind of connectives (*exponentials*, "!" and "?") which shall express the iterability of an action, i.e. the absence of any reaction ; typically $!A$ means to spend as many $'s as one needs. If we use the symbol $\multimap$ (*linear implication*) for causal implication, a usual intuitionistic implication $A \Rightarrow B$ therefore appears as

$$A \Rightarrow B = (!A) \multimap B$$

i.e. A implies B exactly when B is caused by some iteration of A. As far as intuitionistic logic is concerned, the translation inside linear logic using essentially this principle, will be faithful, so nothing will be lost ; it remains to see what is gained.

II.2. the two conjunctions

In linear logic, two conjunctions $\otimes$ (*times*) and & (*with*) coexist. They correspond to two radically different uses of the word "and". Both conjunctions express the availability of two actions ; but in the case of $\otimes$, both will be done, whereas in the case of &, only one of them will be performed (but we shall decide which one). To understand the distinction consider A,B,C :

A :	to spend $1
B :	to get a pack of Camels
C :	to get a pack of Marlboro.

An action of type A will be a way of taking $1 out of one's pocket (there may be several actions of this type since we own several notes). Similarly, there are several packs of Camels at the dealer's, hence there are several actions of type B. An action of type $A \multimap B$ is a way of replacing any specific $ by a specific pack of Camels.

Now, given an action of type $A \multimap B$ and an action of type $A \multimap C$, there will be no way of forming an action of type $A \multimap B \otimes C$, since for $1 you will never get what costs $2 (there will be an action of type $A \otimes A \multimap B \otimes C$, namely getting two packs for $2). However, there will be an action of type $A \multimap B \& C$, namely the superposition of both actions. In order to perform this action, we have first to choose which among the two possible actions we want to perform, and then to do the one selected. This is an exact analogue of the computer instruction IF...THEN...ELSE... : in this familiar case, the parts THEN... and ELSE... are available, but only one of them will be done. A typical misconception is to view "&" as a disjunction : this is wrong since the formulas $A \& B \multimap A$ and $A \& B \multimap B$ will both be provable. By the way, there are two disjunctions in linear logic :

"$\oplus$" (*plus*) which is the dual of "&", which expresses the choice of one action between two possible types ; typically an action of type $A \multimap B \oplus C$ will be to get one pack of Marlboro for the $, another one is to get the pack of Camels. In that case, we can no longer decide which brand of cigarettes we shall get. In terms of computer science, the distinction &/$\oplus$ is reminiscent of the distinction outer/inner non determinism.

"$\parr$" (*par*) which is the dual of "$\otimes$", which expresses a dependency between two types of actions ; the meaning of "$\parr$" is not that easy, let us just say that $A \parr B$ can either be read as $A^{\perp} \multimap B$ or as $B^{\perp} \multimap A$, i.e. "$\parr$" is a symmetric form of "$\multimap$"; if one prefers, "$\parr$" is the constructive contents of classical disjunction.

II.3. states and transitions

A typical consequence of the excessive focusing of logicians on mathematics is that the notion of *state* of a system has been overlooked. Let us give some examples to which the discussion applies :

(i) the current state of a Petri net, i.e. the repartition of tokens (a suggestion of Andrea Asperti and Carl Gunter, private communications)

(ii) the current state of a Turing machine (a suggestion of Vincent Danos, private communication)

(iii) the current position during a chessboard game

(iv) the list of molecules present before (or after) a chemical reaction

(v) the current list of beliefs of an expert system, etc.

Observe that all these cases are modelized according to precise protocols, hence can be formalized, so can eventually be written in mathematics : but in all cases, one will have to introduce an extraneous temporal parameter, and the formalization will explain, in classical logic, how to pass from the state S (modelized as (S,t)) to a new one (modelized as $(S',t+1)$). This is very awkward, and it would be preferable to ignore this *ad hoc* temporal parameter.

In fact, one would like to represent states by formulas, and transitions by means of implications of states, in such a way that S' is accessible from S exactly when $S \multimap S'$ is provable from the transitions, taken as axioms. But here we meet the problem that, with usual logic, the phenomenon of *updating* cannot be represented. For instance take the chemical equation

$$2H_2 + O_2 \Rightarrow 2H_2O$$

a paraphrasis of it in current language could be

$$\text{"}H_2 \text{ and } H_2 \text{ and } O_2 \text{ imply } H_2O \text{ and } H_2O\text{".}$$

The common sense knows how to manipulate this as a logical inference ; but this common sense knows that the sense of "and" here is non idempotent (because the proportions are crucial) and that once the starting state has been used to produce the final one, it cannot be reused. The features which are needed here are those of "$\otimes$" to represent "and" and "$\multimap$" to represent "imply"; a correct representation will therefore be

$$H_2 \otimes H_2 \otimes O_2 \multimap H_2O \otimes H_2O$$

and it turns out that if we take chemical equations written in this way as axioms, then the notion of linear consequence will correspond to the notion of accessible state from an initial one. On this example we see that it is crucial that the two following rules of classical logic

$$A \Rightarrow A \wedge A \qquad (1) \qquad\qquad A \wedge B \Rightarrow A \qquad (2)$$

become wrong when $\Rightarrow$ is replaced by $\multimap$ and $\wedge$ is replaced by $\otimes$ (principle (1) would say that the proportions do not matter, whereas principle (2) would enable us to add an atom of carbon to the left, that would not be present on the right). The first rule is a way of writing *contraction*, whereas the second rule is a way of writing *weakening*.

Let us now go to an example from computer science, which essentially simplifies what Danos did for Turing machines : take a formal grammar, generated by a finite alphabet $L = \langle a_1, \ldots, a_k \rangle$, and defined by means of transition rules

$$m_i \Rightarrow m'_i \qquad (i = 1, \ldots, n)$$

where the m_i's are words on L ; now, if we take $a_1, \ldots, a_k$ as propositional atoms

we can represent any word	$m = a_{i_1} \ldots a_{i_p}$
by means of the proposition	$m* = a_{i_1} \otimes \ldots \otimes a_{i_p}$
and any transition	$m \Rightarrow m'$
by means of the axiom	$m* \multimap m'*$

Then it is plain than a word m is accessible from a word m_0 just in case the linear implication

$$m_0 \multimap m$$

is provable from the axioms. Here together with the interdiction of the analogues of (1) and (2), a third principle from classical logic :

$$A \wedge B \Rightarrow B \wedge A \qquad (3)$$

becomes wrong if one replaces "$\wedge$" by "$\otimes$" and "$\Rightarrow$" by "$\multimap$". (3) is a possible way of writing *exchange*. In that case, we are not longer speaking oɪ standard linear logic, but of *non-commutative* linear logic, which should play a prominent role in the future, but which is still very experimental.

The example of a formal grammar shows very clearly the close connection between linear logic and any kind of computation process, and it is therefore no wonder that linear logic finds natural applications in computer science. Observe also that the meaning of the arrow of state transition systems is now exactly the meaning of a logical implication, what it should be !

To sum up our discussion about states and transitions : the familiar notion of theory : classical logic + axioms, should therefore be replaced by :

theory = linear logic + axioms + current state

The axioms are there forever; but the current state is available for a single use: hence once it has been used to prove another state, then the theory is updated, i.e. this other state becomes the next current state.

II.4. expert systems

Let us say something more specific about expert systems : to cope with the inadequation of classical logic w.r.t. updating, incredible solutions have been proposed, namely to replace theories by models. The authors of such "ameliorations" of classical logic, e.g. the so called "default reasoning" –that would better be called "deficient reasoning"– seem to ignore that processes of the kind "if A cannot be proved, add $\neg$A" have been known for more than 50 years to be non-effective at all. By the way, even the idea of thinking of a decision as a completion process is a nonsense : are we interested to add

"Mary is a student"

to our knowledge on the basis it is consistent ? The answer is negative, since there maybe 300 applicants like Mary (all consistently students) but we have limited *resources*, and we have to select 30 of them by means of some reasonable criteria, money, base-ball etc. Here the real decision problem is of *quantitative* nature (minimal inputs, maximal outputs) and cannot be handled by means of classical logic or any of its "improvements". Here linear logic could be of essential use : for reasons already explained, this logic keeps an exact maintenance of resources ; hence, from the initial state S (representing our resources) it could deduce several possibilities, e.g.

$$S \multimap S'\&S''\&S'''$$

and it is up to us (external non-determinism) to select which among S', S'', and S''' will be our next state. We can even refine the logical axioms in such a way that the choice is directly made during the deduction process, but at no moment we have to go to the shame of completion ! It seems that people were led to such regressions simply because they were not able to give divergent transition rules, e.g. $S \Rightarrow S'$ together with $S \Rightarrow \neg S'$, which in classical logic entail the contradiction of S. In linear logic, one can write simultaneous transitions $S \multimap S'$ and $S \multimap S''$, with S', S'' contradictory, without S becoming contradictory ($S\otimes S$ becomes contradictory, but it has a meaning very far from S).

II.5. linear negation

The most important linear connective is *linear negation* $(-)^\perp$ (*nil*). Since linear implication will eventually be rewritten as $A^\perp \parr B$, "nil" is the only negative operation of logic. Linear negation behaves like transposition in linear algebra ($A \multimap B$ will be the same as $B^\perp \multimap A^\perp$, at least in the commutative case), i.e. it expresses a *duality*, that is a change of standpoint :

$$\textit{action of type } A = \textit{reaction of type } A^{\perp}$$

(other aspects of the duality action/reaction are output/input, or answer/question). This change of standpoint is ultimately an inversion of causality, i.e. of the sense of time, but this aspect is still very mysterious, since it involves non-commutative linear logic, which is not yet ripe.

The main property of $(-)^{\perp}$ is that $A^{\perp\perp}$ can, without any problem, be identified with A, like in classical logic ; but, in classical logic, the price to pay was the loss of constructivity. In intuitionistic logic, it is well known that $\neg\neg A$ is not equivalent with A. But it is less known that the familiar equivalence between $\neg A$ and $\neg\neg\neg A$ is not an isomorphism w.r.t. proofs : we have indeed maps in both directions, which form a retraction pair (like in topological vector spaces, between a dual and a tridual). In fact the familiar Gödel $\neg\neg$-translation of classical logic into intuitionistic logic heavily uses identifications of the form $\neg\neg A = \neg\neg\neg\neg A$, which are wrong in terms of proofs, and this is why this translation is not enough to make classical logic constructive.

The involutive character of "nil" ensures De Morgan-like laws for all connectives and quantifiers, e.g.

$$\exists x A = (\forall x A^{\perp})^{\perp}$$

which may look surprising at first sight, especially if we keep in mind that the existential quantifier of linear logic is *effective* : typically, if one proves $\exists xA$, then one proves $A[t/x]$ for a certain term t. This exceptional behaviour of "nil" comes from the fact that $A^{\perp}$ negates (i.e. reacts to) a single action of type A, whereas usual negation only negates some (unspecified) iteration of A, what usually leads to a Herbrand disjunction of unspecified length, whereas the idea of linear negation is not connected to anything like a Herbrand disjunction. Linear negation is therefore more primitive, but also *stronger* than usual negation (i.e. more difficult to prove).

II.6. structural rules

In 1934 Gentzen introduced *sequent calculus*, which is the basic synthetic tool for studying the laws of logic. This calculus is not always convenient to build proofs, but it is essential to study their properties. (In the same way, Hamilton's equations in mechanics are not very useful to solve practical problems of motion, but they play an essential role when we want to discuss the very principles of mechanics.) Technically speaking, Gentzen introduced *sequents*, i.e. expressions $\Gamma \vdash \Delta$, where Γ $(= A_1,\dots,A_n)$ and Δ $(= B_1,\dots,B_m)$ are finite sequences of formulas. The intended meaning of $\Gamma \vdash \Delta$ is that

$$A_1 \text{ and } \ldots \text{ and } A_n \text{ imply } B_1 \text{ or } \ldots \text{ or } B_m$$

but the sense of "and", "imply", "or" has to be clarified. The calculus is divided into three groups of rules (**identity**, **structural**, **logical**), among which the structural block has been systematically overlooked. In fact, a close inspection shows that the actual meaning of the words "and", "imply", "or", is wholly in the structural group : in fact it is not too excessive to say that a logic is essentially a set of structural rules ! The three standard structural rules are all of the form

$$\frac{\Gamma \vdash \Delta}{\Gamma' \vdash \Delta'} \quad \text{, more precisely :}$$

α) *weakening* opens the door for fake dependencies : in that case Γ' and Δ' are just extensions of the sequences Γ, Δ. Typically, it speaks of causes without effect, e.g. spending \$1 to get nothing -not even smoke-; but is an essential tool in mathematics (from B deduce $A \Rightarrow B$) since it allows us not to use all the hypotheses in a deduction. It will rightly be rejected from linear logic. It is to be remarked that this rule has been criticized a long time ago by philosophers in the tradition of Lewis's "strict implication", and has led to various "relevance logics", which belong to the philosophical side of logic, see [3] for instance. Technically speaking, the rule says that ⊗ is stronger than &, which is wrong, but not that absurd :

$$\text{(I)} \qquad \cfrac{\cfrac{\cfrac{A \vdash A}{A,B \vdash A}\,w \qquad \cfrac{B \vdash B}{A,B \vdash B}\,w}{A,B \vdash A\&B}\,\&}{A\otimes B \vdash A\&B}\,\otimes$$

β) *contraction* is the fingernail of infinity in propositional calculus : it says that what you have, you will always keep, no matter how you use it. The rule corresponds to the case where Γ' and Δ' come from Γ and Δ by identifying several occurences of the same formula (on the same side of "$\vdash$"). To convince yourself that the rule is about infinity (and in fact that without it there is no infinite at all in logic), take the formula INF : $\forall x \exists y\ x < y$ (together with others saying that $<$ is a strict order). This axiom has only infinite models, and we show this by exhibiting 1,2,3,4,... distinct elements ; but, if we want to exhibit 27 distinct elements, we are actually using INF 26 times, and without a principle saying that 26 INF can be contracted into one, we would never make it ! Another infinitary feature of the rule is that it is the only responsible for undecidability : Gentzen's subformula property yields a decision method for predicate calculus, provided we can bound the length of the sequents involved in a cut-free proof, and this is obviously the case in the absence of contraction. This fact has been observed by various people, see e.g. [13] (but there is a lot of reported literature on this subject in Japan and Russia, which may be anterior to the one just mentioned). In linear logic, both contraction and weakening will be forbidden *as structural rules* ; but it would be nonsense not to recover them in some way : we have introduced a new interpretation for the basic notions of logic (actions), but we do not want to abolish the old one (situations), and this is why special connectives (exponentials ! and ?) will be introduced, with the two missing structurals as their main rules. The main difference is that we now *control* in many cases the use of contraction, which, -one should not forget it- means controlling Herbrand disjunctions.

Intuitionistic logic accepts contraction (and weakening as well), but only to the left of sequents : this is done in a very hypocritical way, by restricting the sequents to the case where Δ consists of one formula, so that we are never actually in position to write a single right structural. So, when we have a cut-free proof of $\vdash A$, the last rule must be logical, and this has immediate consequences, e.g. if A is $\exists yB$, then B[t] has been proved for some t etc. These features, that just come from the absence of right contraction, will therefore be present in linear logic, in spite of the presence of an involutive negation. It is perhaps the place to have a discussion on the fuzzy expression "constructive"; what is wrong with classical logic is not that we have *tertium non datur*, since the example of linear logic (namely $A ⅋ A^{\perp}$) shows that it can very well be interpreted as a communication : on the whole any interpretation of classical disjunction in terms of operations on proofs would be admissible. Such operations are in fact defined via cut-elimination, which precisely gives the meaning of proofs as functions. But, in the case of a cut

$$\dfrac{\dfrac{\Gamma \vdash A,A,\Delta}{\Gamma \vdash A,\Delta} \qquad \dfrac{\Gamma',A,A \vdash \Delta'}{\Gamma',A \vdash \Delta'}}{\Gamma,\Gamma' \vdash \Delta,\Delta'}$$

between a right and a left contraction, Gentzen's procedure yields two essentially different answers, depending on which side we take as the most important : in other terms a symmetric problem gets several asymetric answers. This example shows why classically speaking, all proofs must be identified, that is why classical logic is eventually about *provability*, and not about proofs (in terms of categories, there is a similar remark by Joyal, see [15], pp. 65-67, 126. Hence the ultimate reason why classical logic is not constructive is not because it uses contraction on the right, but because it uses it on *both* sides.

Relevance logic accepts contraction on both sides, and removes weakening ; the result of this cocktail of structural rules is very awkward since it seems that the good combinations are : C+W+E (classical), W+E (affine), E (linear), nothing (linear non commutative). The awkwardness of the logic is made even worse by the adjunction of *ad hoc* distributivity rules, which come from an attempt to stick -as much as possible in the absence of weakening- to classical logic. In terms of resources, relevantists correctly stated that the premise must be used in a causality, but their acceptation of contraction now says that resources may be used *ad libitum* : from two pieces of bread you will never go to one without eating, but from one, you can get 1000, like in Jesus's miracles... The fact that they kept contraction on both sides left the problem of constructivity untouched. In fact, if we clearly lose something with relevantism (namely the simplicity, the elegance of classical logic), it is hard to say what is gained since relevantism is not constructive, and just corresponds to vague philosophical motivations. However, to be fair, relevantists were the first people to distinguish between two conjunctions, two disjunctions, with exactly the rules that we later wrote for ⊗, ⅋, &, ⊕. In terms of formulation of the rules, the distinction ⊗/⅋ is already legitimate in classical logic, but has been overlooked, since these connectives are provably equivalent (this is the reason why the distinction is more natural in logics that do not contain weakening and/or contraction). The fact that "⊗" is stronger than "&" comes from weakening (see (I) above) ; using contraction, we get the reverse implication :

$$\text{(II)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{A \vdash A \qquad B \vdash B}{A, B \vdash A \otimes B}\otimes}{A\&B, B \vdash A \otimes B}\&}{A\&B, A\&B \vdash A \otimes B}\&}{A\&B \vdash A \otimes B}c$$

This proof -available in relevance logic too- says that "&" is stronger than "⊗", which is much worse than the other side of the equivalence proved by (I) above. It makes the distinction between ⊗ and & very tiny, and on the whole useless. If we now look at classical logic, we see that the main meaning of "$\wedge$" and "$\vee$" as *connectives* is additive ($\wedge = \&$, $\vee = \oplus$), whereas the meaning of "$\wedge$","$\vee$" as *commas* in sequents, is multiplicative ($\wedge = \otimes$, $\vee =$ ⅋). Classical logic -and to a large extent relevance logic- operates a confusion between additive and multiplicative features, and these features dislike extremely to be confused.

γ) *exchange* expresses the commutativity of multiplicatives : Γ' and Δ' are obtained from Γ and Δ by inner permutations of formulas. It is only for reasons of expressive power that this rule is still present in the main version of linear logic : a certain amount of commutativity is needed in order to make a good use of exponentials. Here one has to mention the work of Lambek (1958) [14], which came out from linguistic considerations; his *syntactic calculus* is based on a non-commutative conjunction (corresponding to our $\otimes$) and two implications, $\multimap$ and $\multimapinv$, one to the right and one to the left. The general framework is that of intuitionistic sequents $\Gamma \vdash A$, with no structurals at all. In spite of its limited expressive power (only multiplicatives) and the artificial intuitionistic framework, this work must be acknowledged as a true ancestor to linear logic ; its connection to linguistics can be seen as the first serious evidence against the exclusive focus on mathematics. Moreover, its rejection of exchange seem to indicate that, eventually, linear logic should be non-commutative, i.e. without exchange. See II.9. for a discussion of non-commutativity.

As soon as weakening and contraction have been expelled, one can imagine other structural rules, among which the *mix* rule

$$\frac{\Gamma \vdash \Delta \qquad \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

has some interest. If one were to accept this rule, then good taste would require to add the void sequent $\vdash$ as an axiom (without weakening, this has no dramatic consequence). *Mix* is connected with the neutral multiplicative elements $\mathbf{1}$ and $\perp$, more precisely, it states that they are the same, which is of course a simplifying hypothesis. It can also be viewed as the fact that $\otimes$ is stronger than $⅋$, which means that the absence of interaction should be a form of interaction. As a matter of fact there is not enough material to make a definite judgement as to a possible inclusion of *mix*. For the reader acquainted with proof-nets [7], the inclusion of this rule would have an unpleasant feature, namely to abolish the idea of *cyclicity* which enabled us to forward the information from any part of the structures to any other part.

II.7. linear sequent calculus

In order to present the calculus, we shall adopt the following notational simplification : formulas are written from literals $p, q, r, p^\perp, q^\perp, r^\perp$ etc. and constants $\mathbf{1}$, $\perp$, $\top$, $\mathbf{0}$, by means of the connectives $\otimes$, $⅋$, $\&$, $\oplus$ (binary) $!$, $?$ (unary) and the quantifiers $\forall x$, $\exists x$. Negation is *defined* by De Morgan equations , and linear implication is also a defined connective :

$1^\perp =_d \perp$	$\perp^\perp =_d 1$
$\top^\perp =_d 0$	$0^\perp =_d \top$
$p^\perp =_d p^\perp$	$p^{\perp\perp} =_d p$
$(A\otimes B)^\perp =_d A^\perp ⅋ B^\perp$	$(A⅋B)^\perp =_d A^\perp\otimes B^\perp$
$(A\& B)^\perp =_d A^\perp\oplus B^\perp$	$(A\oplus B)^\perp =_d A^\perp\& B^\perp$
$(!A)^\perp =_d ?A^\perp$	$(?A)^\perp =_d !A^\perp$
$(\forall xA)^\perp =_d \exists xA^\perp$	$(\exists xA)^\perp =_d \forall xA^\perp$

$$A \multimap B =_d A^\perp ⅋ B$$

Sequents are now of the form $\vdash \Delta$, i.e. the left hand side is void. General sequents $\Gamma \vdash \Delta$ can be mimicked as $\vdash \Gamma^\perp,\Delta$.

IDENTITY GROUP

$\vdash A,A^\perp$ (Identity axiom)

$$\frac{\vdash \Gamma,A \qquad \vdash A^\perp,\Delta}{\vdash \Gamma,\Delta}\ (\mathit{Cut}) \qquad\qquad \text{(Cut rule)}$$

STRUCTURAL GROUP

$$\frac{\vdash \Gamma}{\vdash \Gamma'}\ (e) \qquad\qquad \text{(exchange)}$$

in this rule Γ' is obtained from Γ by a permutation.

LOGICAL GROUP

multiplicatives

$$\frac{\vdash \Gamma,A \qquad \vdash B,\Delta}{\vdash \Gamma,A\otimes B,\Delta}\ (\otimes) \qquad\qquad \frac{\vdash \Gamma,A,B}{\vdash \Gamma,A⅋B}\ (⅋)$$

$$\vdash 1 \quad \text{(axiom)} \qquad\qquad \frac{\vdash \Gamma}{\vdash \Gamma,\perp}\ (\perp)$$

additives

$$\frac{\vdash \Gamma,A \qquad \vdash \Gamma,B}{\vdash \Gamma,A\& B}\ (\&) \qquad\qquad \frac{\vdash \Gamma,A}{\vdash \Gamma,A\oplus B}\ (\oplus^1) \qquad \frac{\vdash \Gamma,B}{\vdash \Gamma,A\oplus B}\ (\oplus^2)$$

$\vdash \Gamma,\top$ (axiom)

exponentials

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ (!) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ (d?) \quad \text{(dereliction)}$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ (w?) \quad \text{(weakening)}$$

$$\frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ (c?) \quad \text{(contraction)}$$

quantifiers

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall x A}\ (\forall) \qquad \frac{\vdash \Gamma, A[t/x]}{\vdash \Gamma, \exists x A}\ (\exists)$$

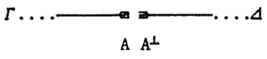
(In rule ($\forall$), x must not be free in Γ)

II.8. comments

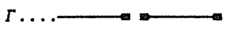
The rule for "$⅋$" shows that the comma behaves like a hypocritical "$⅋$" (on the left it would behave like "$\otimes$"); "and", "or", "imply" are therefore read as "$\otimes$", "$⅋$", "$\multimap$".

identity group : the principles of this group express that "A is A", which is perhaps the ultimate meaning of logic ... In other terms, they say that an action (output, answer) of type A is a reaction (input, question) of type $A^\perp$. One can also view the identity axiom as the identity function from A to A, or from $A^\perp$ to $A^\perp$, and the symmetry of the axiom forbids us to choose between these two legitimate interpretations. But in fact, the interpretation as a function is wrong, since it forgets the dynamics. Let us try to understand this very important point : the only dynamical feature of the system is the cut-rule ; without cut, there would be no action performed. The cut puts together an action and a reaction of the same type (i.e. an action of type A and an action of type $A^\perp$) and something happens, namely "cut elimination". Let us take a very trivial analogy, namely DIN plugs for electronic equipment : we may think of the axiom $\vdash A, A^\perp$ as an extension wire between two complementary DIN plugs :

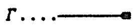
More generally, we can think of a sequent Γ as the interface of an electronic equipment, this interface being made of DIN plugs of various forms ; the negation corresponds to the complementarity between male and female plugs. Now a proof of Γ can be seen as any equipment with interface Γ. Now, the cut rule is well explained as a plugging :



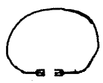
the main property of the extension wire is that



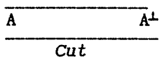
can be replaced by



It seems that the ultimate, deep meaning of cut-elimination is located there. Observe that commonsense would forbid self-plugging of an extension wire :



which would correspond, in terms of the proof-nets of [7] to the incestuous configuration :



which admits two shortrips.

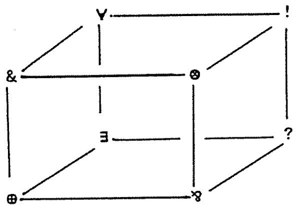
structural group : see additional discussion in II.9. .

logical group :

multiplicatives and additives : notice the difference between the rule for ⊗ and the rule for & : ⊗ requires disjoint contexts (which will never be identified unless ? is heavily used) whereas & works with twice the same context. In a similar way, the two disjunctions are very different, since ⊕ requires one among the premises, whereas ⅋ requires both).

exponentials : ! and ? are modalities. Modalities are very special connectives. The rule for ! does not define this connective, since the context Γ must start with "?", which is the dual of "!", i.e. "!" is eventually defined in terms of itself. To understand the difference between exponentials and -say- additives, let us remark that, if we write additive rules for another pair &', ⊕', then the equivalence of this new pair with &, ⊕ is immediate, whereas, one can have another pair !', ?', with the same rules as !, ?, but non provably equivalent. The general symmetries of sequent calculus (namely that a cut on a complex formula splits into simpler cuts, and the same for axioms) are such that, when

we know the (right) rule for &, then the rules for $\oplus$ are forced, and conversely. In the case of modalities, the rule for ! does not determine the rules for ? (except dereliction), in particular, the possibility of refining the exponentials is widely open for that reason. The connective "!" (*of course*) has the meaning of a storage. To be very precise, "!" indicates the *potentiality* of a duplication : an action of type !A consists of an action of type A on the slot of a copying machine. The rules for "?" (*why not*) enable us, via cut, to make this machine work : dereliction just takes back the original action, weakening destroys it, while contraction duplicates the data. In terms of computer, the rules for ! and ? correspond to storing, reading, erasing and duplicating. The importance of exponentials w.r.t. memory has been pointed out by Yves Lafont : see [11], in particular the idea of computing without garbage collector.
quantifiers : they are not very different from what they are in usual logic, if we except the disturbing fact that $\exists$ is now the exact dual of $\forall$. It is important to remark that $\forall$ is very close to & (and that $\exists$ is very close to $\oplus$). But are there quantifier analogues for "$\otimes$" and "$\parr$" ? Clearly such "multiplicative quantifiers" should differ from $\forall$ and $\exists$ in the sense that they would allow not a single instanciation, but several simultaneous instanciations, and therefore they would be very close to exponentials. In other terms, one cannot exclude the replacement of the awkward modals "!" and "?" by some more regular quantifiers. The eight logical operations can be written on a cubic pattern :



organized along the three oppositions

vertical : conjunctive/disjunctive

horizontal : one/all

oblique : binary/uniform.

II.9. non-commutativity

Non-commutative linear logic is still very experimental, and we shall here just discuss the most obvious system, which contains no exponentials. Compared to II.7., we must make the following adaptations :

$(A\otimes B)^{\perp} =_d B^{\perp}\parr A^{\perp}$ $(A\parr B)^{\perp} =_d B^{\perp}\otimes A^{\perp}$

Besides linear implication

$A \multimap B =_d A^\perp ⅋ B$

there will coexist linear *retro*-implication,

$B \multimapinv A =_d B ⅋ A^\perp$

Transposition will no longer be the identification between $A \multimap B$ and $B^\perp \multimap A^\perp$ (which involved commutativity of ⅋), but the fact that $A \multimap B$ and $A^\perp \multimapinv B^\perp$ are literally the same. One of the most exciting (and problematic) intuitions about non-commutativity is that it should have a temporal meaning : $\multimap$ is usual causality (in the future), whereas $\multimapinv$ is past causality. Everybody will easily find examples of past causality in real life; however since we are not accustomed to think rigorously in those terms, the most basic evidences may be misleading, and the existence of some formal model (even very incomplete) might be of essential use. This is to say that in such matters, we cannot use common sense at all, and that we must follow some kind of mathematical pattern, even against intuition. Here comes a very interesting phenomenon (that can be understood by those who know proof-nets, see [7]) : the natural way of introducing non-commutativity is not to expell exchange, but to restrict it to circular permutations. In fact this corresponds exactly to the restriction to *planar* proof-nets, i.e. to proof-nets with no crossings between the axiom links. But if we start to forbid crossings of lines in the non-commutative case, do this mean that the commutative proofs are incorrect ? Or should we, as suggested by Freyd (private discussion) view them as *proof-braids* ? Surely linear logic has a lot of relations with monoidal categories (there are even categories like Barr's "*-Autonomous categories" [1] which have additives, multiplicatives and an involution), and proof-braids could be a good candidate to describe the various isomorphisms in a non-commutative monoidal category. But what could be the temporal meaning of the twistings between axiom links ? This seems to be an absolute mystery ...

The restriction to circular permutations (which means that we consider the sequents as written on a circle) makes a reasonable candidate for a logic (as long as we ignore exponentials). In view of the identification between

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

which is essentially associativity of "$\otimes$", we can even understand that in the product $B \otimes A$, the second component is done before the first one.

We can keep the rules as they have been written in II.7. ; if we want to have 2-sided sequents, it is natural to translate $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ as $\vdash A_n^\perp, \ldots, A_1^\perp, B_1, \ldots, B_m$. One of the obvious features of the calculus is that the two implications do not mingle, and this will be essential when we shall try to work with PROLOG.

II.10. logic programming

It seems that the exact relation of PROLOG (and more generally Robinson's resolution) to Gentzen's *Hauptsatz* has been overlooked in the full literature on logic programming, so let us explain this roughly : the famous result of Gentzen states that, if $\vdash A$ is a logical consequence (in the sense of classical logic) of axioms $\Gamma_i \vdash \Delta_i$, then there is a proof of A, with the following properties :

it uses as axioms instanciations $\Gamma'_i \vdash \Delta'_i$ of the original axioms

the cut rule is restricted to cut-formulas occuring in some instanciation of those axioms.

(For a textbook presentation of this result, see e.g. [6], p.123.)

This is not exactly Robinson's resolution, but it does not take much to get it from that result : assume now that the axioms are Horn clauses, i.e. Γ_i and Δ_i are made of atomic formulas, and Δ_i is just one formula ; assume also that A is atomic. Then since cut is restricted to atomic formulas, logical rules cannot be used (because logical symbols cannot disappear), and we are therefore left with the following list of principles :

i) instanciations of axioms (it is enough to consider non-logical axioms)

ii) cut

iii) weakening

iv) contraction

v) exchange

It is easily shown that one can restrict oneself to deductions made only of Horn clauses ; then one gets rid of :

iii) weakening : if we replace some clause $\Gamma \vdash C$ by $\Gamma, B \vdash C$, since our goal has no left part, the fate of B is to disappear by means of a cut, and we can assume that this will happen by proving at some later moment $\vdash B$: but this is masochism ! It would have been simpler not to weaken at all.

iv) contraction : if we replace some clause $\Gamma, B, B \vdash C$ by $\Gamma, B \vdash C$, since at some moment, our B must be cut with one proof of $\vdash B$ (as above), it would have been possible to cut twice at the level of $\Gamma, B, B \vdash C$ to get $\Gamma \vdash C$. Observe that this idea removes a rule, but the price to pay is a doubling of the task, since B will be searched for twice.

v) exchange : it is finally possible to do our cuts in such an order that exchange is never used.

Finally, we are left with steps i) and ii), and the familiar resolution method is nothing but automatic proving inside i) + ii). But this means that w are in fact trying to make proofs inside non-commutative linear logic ! Now it is necessary to make a very important -although commonplace- remark : so-called logic programming is logical at two levels

externally, since it is concerned with the notion of classical consequence

internally, since its operationality obeys to certain logical laws (which indeed are those of non-commutative linear logic)

The two logics, the internal and the external one, have no reason to coincide ; it is because people want them to coincide that they are lead to absolute nonsense. Let us take an example : in classical logic (which is the external logic of PROLOG), everybody knows that there are 16 binary connectives, period. A quick inspection of the list shows that only one of them looks like a conjunction : this is the familiar boolean conjunction which is desperately commutative and idempotent. But is the inner conjunction of PROLOG like that ? If we compare the clauses $A,A \vdash B$ and $A \vdash B$, it is very clear that the first one requires a duplication of the intermediate goal A, hence *internally* speaking, "A and A" does not mean "A", unless we decide to ignore problems of efficiency.

But the inner conjunction is not even commutative, since the clauses $A,B \vdash C$ and $B,A \vdash C$ have a very different behaviour, typically when B fails and A neither succeeds nor fails ; the failure of commutativity is by the way much more extreme than the failure of idempotency. Since classical logic implies the commutativity of conjunction, the only way to avoid an immediate contradiction in the example just given is to identify failure with the absence of success, i.e. to introduce unprovability, which leads to the deficient nonsense already met in section II.4. : this is the so-called "closed world assumption", which came from the furious attempt to identify inner and outer logic, and just succeeded in writing absurdities both from the viewpoint of classical logic and from the viewpoint of operationality.

But failure is not the absence of success : failure is a positive information, namely that we have followed a certain operational pattern *up to the end*, and that we are aware of it. This is definitely different from the absence of success which could for instance involve non terminating loops. So, if we can express exactly which operational pattern we have followed, it will be possible to internalize failure as the provability of some statement, and since provability is the kind of feature that can be mechanized, we are not led to computational nonsense, like in the case of the "closed world assumption". In particular, the idea of "negation as failure" is perfectly sound, provided we are prepared to accept that such a negation cannot at the same time be classical ! In fact we propose to identify negation in PROLOG with linear negation, then to use the usual implication "$\rightarrow$" to speak of success, and the reverse arrow "$\leftarrow$" to speak of failure. Let us give a very primitive example : we just consider propositional clauses A,B,C etc. and we assume that there are only two clauses ending with E, namely

(I) $A,B \vdash E$

(II) $C,D \vdash E$

Let us describe in familiar terms the part of the procedure involving the subgoal E : we try to prove E by means of (I), and in case attempt (I) fails, we try by means of (II); if (II) fails as well, then E fails. To say that attempt (I) succeeds is just $A \otimes B$ ("do B, then do A"), and the fact that success is then forwarded to E is therefore

(1) $A \otimes B \multimap E$

Failure of attempt (I) can be decomposed into two subcases : either B fails, or B succeeds and then A fails. This will be exactly represented by $B^\perp \oplus (A^\perp \otimes B)$; now, once attempt (I) has failed, a success of attempt (II) will be forwarded to E :

$C \otimes D \otimes (B^\perp \oplus (A^\perp \otimes B)) \multimap E$

which can be written as two axioms

(2) $C \otimes D \otimes B^\perp \multimap E$

(3) $C \otimes D \otimes A^\perp \otimes B \multimap E$

Finally, failure of both (I) and (II) will cause a failure of E :

$(D^\perp \oplus (C^\perp \otimes D)) \otimes (B^\perp \oplus (A^\perp \otimes B)) \multimap E^\perp$

which can be written as

(4) $D^\perp \otimes B^\perp \multimap E^\perp$

(5) $D^\perp \otimes A^\perp \otimes B \multimap E^\perp$

(6) $C^\perp \otimes D \otimes B^\perp \multimap E^\perp$

(7) $C^\perp \otimes D \otimes A^\perp \otimes B \multimap E^\perp$

Axioms (4)-(7) are indeed retro-causalities "in order to do E, one must do...", Typically (4) is $B \parr D \multimapinv E$, etc. In fact, (1)-(7) can be written as right-handed sequents made of literals, typically (5) becomes

(5') $\vdash B^\perp, A, D, E^\perp$ etc.

Then one should now compare this axiomatization with the operationality of PROLOG, and that task should better be done by specialists of logic programming (by the way, there is some work in preparation with Jean Gallier and Stan Raatz,

whose goal is the study of any form of control in logic programming by means of a linear logic axiomatization).
Let us just mention some key points :
i) it is very important that the two implications do not mingle ; cyclic exchange may cause some problems, easily solved by adding two special constants e, s, with $e^{\perp} =_d s$, $s^{\perp} =_d e$, and writing, instead of (5') :

(5") $\vdash e, B^{\perp}, A, D, E^{\perp}\otimes s$

and then looking at the provability of goals

$\vdash e, E\otimes s$ or $\vdash e, E^{\perp}\otimes s$

to express success or failure.
ii) a careful look shows that we are indeed using several meanings of "and", "or", "implies". Typically, "$\oplus$" is used when we are listing several possibilities (e.g. for failure), whereas "$⅋$" occurs by means of retrocausality and indicates a complex interaction between subgoals.
<u>II.11. relation with intuitionistic logic</u>
There is a translation of intuitionistic logic inside linear logic : read

$A \Rightarrow B$	as	$!A \multimap B$
$A \wedge B$		$A \& B$
$A \vee B$		$!A \oplus !B$
$\forall xA$		$\forall xA$
$\exists xA$		$\exists x!A$
$\neg A$		$!A \multimap \mathbf{0}$

see [7], ch. 5 for more details. This translation is faithful not only w.r.t. provability, but also w.r.t. proofs. As a matter of fact, linear logic came from a denotational decomposition of disjunction which involved linearization processes, and later on gave rise to all the connectives of linear logic. But an essential concern has always been the possibility of a faithful translation of intuitionistic logic, since we were afraid of a possible loss of expressive power : this is why non-commutative linear logic was a bit overlooked in the beginning.

To give a demonstration of the improvements immediately caused by this translation, let us look at the familiar technique of *fake substitution* in typed λ-calculus : in order to improve implementations, people imagined to indicate substitutions, but not to make them : between $(\lambda xt)u$ and $t[u/x]$, they have introduced an intermediate step, namely $t\langle u/x\rangle$, where $.\langle ./x\rangle$ stands for a fake substitution. This operation has no logical status, which makes its manipulation dangerous. In linear logic, this intermediate step will be built in : since $A \Rightarrow B$ is $!A \multimap B$, we need two steps to mimick a λ-abstraction, so to speak

(1) a "memory" step

(2) a "linear λ-abstraction"

when we normalize, usual β-conversion splits into two steps, one dealing with (2), the other with (1), and these two steps altogether mimick β-conversion. But if one stops at step (2), then one gets something that could be denoted by $t\langle u/x\rangle$, and this eventually gives a logical status to what was originally just control.

III. the main methodological conflicts

III.1. against reductionism

The first important methodological contradiction lies in the oppositions

dynamic	/	*static*
sense	/	*denotation*
finite	/	*infinite*

these three oppositions are different aspect of the same problem.

Let us start with Frege, who distinguished between *sense* and *denotation* : if we take the sentence

Erich von Stroheim is the author of *Greed*

"Erich von Stroheim" and "the author of *Greed*" have the same denotation, i.e. represent the same external object, but have not the same sense (otherwise it would be pointless to state such a sentence). Denotationally speaking the two expressions refer to the same thing, whereas one has to check something (look at a dictionary, make a proof, a computation) to relate their two distinct senses. This is why

denotation is static, sense is dynamic.

The only extant mathematical semantics for computation are denotational, i.e. static. This is the case for the original semantics of Scott [17], which dates back to 1969, and this remains true for the more recent *coherent semantics* of the author [7]. These semantics interpret proofs as functions, instead of actions. But computation is a dynamic process, analogous to -say- mechanics. The denotational approach to computation is to computer science what statics is to mechanics : a small part of the subject, but a relevant one. The fact that denotational semantics is kept constant during a computational process should be compared to the existence of static invariants like mass in classical mechanics. But the core of mechanics is dynamics, where other invariants of a dynamical nature, like energy, impulsion etc. play a prominent role. Trying to modelize programs as actions is therefore trying to fill the most obvious gap in the theory. There is no appropriate extant name for what we are aiming at : the name "operational semantics" has been already widely used to speak of step-by-step paraphrases of computational processes, while we are clearly aiming at a less *ad hoc* description. This is why we propose the name

geometry of interactions

for such a thing.

The inadequation of the denotational approach w.r.t. computation becomes conspicuous if we observe that such semantics will have a strong tendency to be infinite, whereas programs are finite dynamical processes. So to speak, the denotational approach exchanges a finite dynamical action for an infinite static situation. How is it possible ? Simply by considering not only the behaviour of the system in an actual run (which is very difficult to analyze), but taking at the same time all possible behaviours ; typically, if a program is functional, by listing, in front of every possible input, the corresponding output. This yields an inifinitary expansion, in which the results are flatly embedded. Let us take a very basic example : a program of type

$$(A_1\&B_1)\otimes\ldots\otimes(A_n\&B_n)$$

will be run by choosing between A_1 and B_1, ..., A_n and B_n. The denotational approach will consider altogether the 2^n possible runs, so to include the actual one. But only one of these runs is actual, and this interpretation is clearly wrong. Remark here that syntax is much less greedy, since it encodes the situation with only 2n data. Another familiar example of replacement of a finite dynamical process by an infinite listing is the well-known "ω-rule", prominent in German proof-theory : since the dynamics of induction (i.e. recurrence) is very difficult to handle, one introduces flat listings A[0],A[1],A[2],....; then one has to cope with infinitary syntax, which means that eventually one has to encode it by means of -say- Kleene brackets to come back to the finite. This would not be so bad if the dynamics of Kleene indices were not so *ad hoc*. One has eventually exchanged an intrinsic dynamics for an *ad hoc* one, and something essential has been lost. This is why there is little room for the ω-rule in computer science. It is not absurd to dream of a direct dynamical approach to induction, where the infinite proof-tree would only be an ideal (direct) limit that we never reach : but this is not that easy...

Hilbert's mistake, when he tried to express the infinite in terms of the finite was of a reductionist nature : he neglected the dynamics. The dynamics coming from the elimination of infinity is so complex that one can hardly see any reduction there. But once reductonism has been dumped, Hilbert's claim becomes reasonable : infinity is an undirect way to speak of the finite ; more precisely infinity is about finite dynamical processes.

These basic oppositions are at work when we try to understand "!" ; should we think of !A as something like an infinite tensor $\bigotimes_\omega A$? With some minor adjustments, e.g. $\bigotimes_\omega (1\&A)$, this is denotationally sound ; however, such an identification would be a complete dynamical nonsense. This is because the rule for "!" indicates unlimited *possibilities* of duplication, but not a concrete one : the duplication occurs during elimination of cuts with $?A^\perp$, and it is in this dual part that the information "how many copies do you want" is located. We must not confuse a copying machine that can produce 3000 copies of an original document with these 3000 copies : maybe we only need one. The clarification of this point could be of great importance : consider for instance *bounded exponentials* $!\alpha A, ?\alpha A$, that could be added to linear logic with the intuitive meaning of "iterate α times". They obviously obey to the following laws :

$$\frac{\vdash ?\gamma\Gamma, A}{\vdash ?\alpha\gamma\Gamma, !\alpha A}(!) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?1A}(d?) \quad \text{(dereliction)}$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?0A}(w?) \quad \text{(weakening)}$$

$$\frac{\vdash \Gamma, ?\alpha A, ?\beta A}{\vdash \Gamma, ?\alpha{+}\beta A}(c?) \quad \text{(contraction)}$$

and this shows that there is some underlying polynomial structure in the exponentials. Now, it is not always the case that we can associate polynomials to all exponentials occuring in a proof of standard linear logic, especially when we have to deal with cut ; hence the proofs admitting such polynomial indexings are very peculiar. In fact they admit a normalization in polynomial time : it has already been observed in [7], that, without contraction, the cut-elimination process is essentially shrinking (hence in linear time). Now, if we are with polynomial exponentials, what we can do is first replace all $!\alpha A$ (resp. $?\alpha A$) by something like $\bigotimes_\alpha A$ (resp. $\bigparr_\alpha A$), which induces a polynomial expansion of the proof, and then normalize. Some experimental refinements of linear logic -in order to cope with the converse problem- are under study (with André Scedrov and Phil Scott) and it seems likely that polynomial time functions are typable in such systems. Unfortunately what has so far been done is syntactically awkward, whereas the idea clearly deserves a natural syntax.

IV.2. against subjectivism

Finally, what is closest to the idea of dynamics is syntax, which makes all the necessary distinctions of sense and has the good taste of being finite. So why not contenting oneself with syntax ? This leads to our second opposition :

geometry / taxonomy

In fact this opposition is much more central than the former ones. The problem with syntax is that it is good in many respects but one : it contains irrelevant informations. These informations are very often of temporal nature, and induce an *ad hoc* temporality. Our problem will be to find out what is hidden behind syntax, without going to denotation, so to speak :

a non-subjectivistic approach to sense.

To understand how syntax may convey artificial information, imagine that I want to write something like $A\otimes B\otimes C$. Since $\otimes$ is binary, I must choose between $(A\otimes B)\otimes C$ and $A\otimes(B\otimes C)$, whereas I had in mind a ternary construction. So both solutions contain an irrelevant information, namely some *ad hoc* temporality in the building of $A\otimes B\otimes C$. For instance if I have chosen the first representation, then the subconfiguration $A\otimes B$ will be easier to handle than $B\otimes C$.

α) *what is taxonomy ?*

Taxonomy is the habit of classification by means of dictionaries, languages etc. It is an essential human activity, and corresponds to our need for putting some order into the reality we are dealing with. Let us mention :

i) the *classification of animals* into various species, subspecies etc.
ii) Mendelejeff's *periodic classification* of elements
iii) the use of *coordinates* on Earth
iv) *musical notation*

These four examples are very useful in real life, but not of the same quality. For instance, there is something arbitrary in iii) (e.g. the starting meridian, the units), but it stays reasonable as long as the coordinates are organized along the rotating axis of Earth ; iv) is clearly the product of historical accidents, which have eventually produced 10-odd keys, and is clearly an obstacle to musical practice ; i) is very *ad hoc* : one counts numbers of eyes, wings, testicles, but the discovery of Australia forced taxonomists to add new entries to the classification ; ii) is very good, because based on the number of protons of atoms ; it is so good that new elements were found (or even created) by looking at the gaps of the table, which would have been impossible if these elements had been classified in the style of i) by means of shapes, colours, odours etc. To understand the difference between the scientific quality of i) and ii) : nowadays, we can exclude the existence of a fabulous metal like orichalch, but not of a fabulous beast like a unicorn.

Taxonomy is the bureaucracy of science. This means that -like its State analogue- it is useful, but has a propensity to produce endless obstacles to the execution of the simplest task : its natural tendency is to live on its own and to expell the reality it is about. Taxonomy can be that bad because of the human need to name, to classify, to number, by means of anything whatsoever, Zodiacal signs, "psychological" tests etc.

β) *progress in science* is very often connected with a fight against taxonomy : typically, the discovery that the distinction parabola / hyperbola / ellipse is irrelevant from the algebraic viewpoint, and the subsequent introduction of the projective plane. Algebraically speaking the distinction between these three types of curves appears as a taxonomy, namely the choice of points at infinity. The usual way of getting rid of taxonomy is by exhibiting some kind of *invariants* w.r.t. a natural notion of equivalence, e.g. homotopy etc. The determination of the kind of equivalence depends on scientific choices, e.g. we decide to focus on algebraic properties, and to forget metric ones.

γ) *mathematical logic*, which deals by definition with language, is often taking the opposite viewpoint, namely that any mathematical object comes with an *extrinsic* description, which is claimed to be part of its structure. This viewpoint is widely spread under the name "intensionality". On the whole, the tendency of intensionality would be to distinguish between right and left-handed cups, because we have in mind different uses (which hand holds the cup), whereas mathematicians (and manufacturers) will identify the two things. However the positive aspect of intensionality is that it has kept alive distinctions like

functions as graphs / *functions as programs*

which are surely very important, and that have been overlooked by the main stream of mathematics -especially the Cantorian approach-. The negative aspect of this tradition lies in its implicit slogan "*the map is the territory*", which strongly opposes to any serious structural study.

δ) *recursive functions* are a typical example of this situation. We have been knowing for more than 50 years that a recursive function is not a graph, but a finite program. Kleene indices introduce a class of finite programs which is enough to make decent study of recursive functions, together with nice tools -like the recursion theorem, the S_{nm} theorem- which enable us to manipulate them in a uniform way. Kleene indices convey the necessary amount of finiteness, dynamicity etc. needed to *speak* of programs, *but they are not programs*. They are just some *ad hoc* taxonomy enabling us to reduce an actual program to something which -roughly speaking- does the same thing. The main stream of recursion theory treats Kleene brackets as a black box, enjoying some abstract enumeration property... *But who has ever seen the tail of an actual index ?*

There should be somewhere a purely geometrical notion of finite dynamical structure (not relying on *ad hoc* schemes and dirty encodings). The problem is to find tools sharp enough to isolate them. Our methodological hypothesis is that the problem is *implicitely* solved by Gentzen's *Hauptsatz*, which eliminates abstract notions in proofs (cuts) by introducing finite dynamics, and therefore looks like a good approximation to a universal dynamics. To solve the problem explicitly would mean to find out the geometrical meaning of the *Hauptsatz*, i.e. what is hidden behind the somewhat boring syntactical manipulations it involves.

On this precise point (to take the *Hauptsatz* as our ultimate reference) we shall certainly disagree with category-theorists ; besides personal taste, one must acknowledge that category-theory presents a very clean approach to many problems of computer-science. Unfortunately (if one forgets hypothetical uses of bicategories) it is purely denotational, i.e. modelizes computation by equalities. However one can distinguish between *dynamic* and *static* uses of equality in the categorical approach : typically when we formulate a universal problem, we write a commuting diagram, together with unicity requirements. The commuting diagrams are enough for computing, so the equality here has a dynamic meaning ; the unicity requirement is essentially about possible transpositions of rules, hence is just a change of description (taxonomy), and the equality there is static. See the introduction of [9] for a short discussion.

IV. the geometry of cut-elimination

IV.1. system F

This system, also known as polymorphic λ-calculus, is built as the Howard-isomorphic copy of the system of natural deduction for second order propositional logic. Due to the Howard isomorphism (which is a precise technical restatement of Heyting's paradigm) it will be enough to concentrate on the aspect "natural deduction" of the system. The main features of the system have been established in [5] :

i) normalization : one can execute the programs represented by the proofs by means of certain rewritings that do converge

ii) representation : any numerical algorithm that has a proof of termination inside usual mathematics (i.e. : second order arithmetic) can be represented inside the system. However the internalized algorithm will be slightly different from the original one. Therefore we expect that, when we express some provably terminating algorithm inside **F**, *some regularity is added.*

The first approximation to a universal geometry of computation could be the rewriting process i), but this is wrong : this process has in turn to be implemented. This implementation should not be left to engineering : the mathematical study of what is behind rewriting is a more technical expression of our program of "geometry of interaction".

In what follows, a complete familiarity with Howard's isomorphism [12] (see also [10]) is supposed. This isomorphism enables us to replace functional expressions by deductions. The gain is that, when we view functional terms as proofs, there are some tortures that we can inflict to them that we could not perform on functional expressions : in particular, the symmetrization I/O that will eventually lead to proof-nets. Many misconceptions concerning linear logic and proof-nets come from the fact that people stick too much to the idea of variable. But the use of variables must be seen as a taxonomy : we follow too much our old-fashioned intuitions about inputs and outputs and in particular the functional notation asymmetrizes situations which are perhaps symmetric (again think of right and left handed cups).

IV.2. natural deduction

Natural deduction has mainly been studied by Dag Prawitz in the 60's, see [16], [10]. It can be seen as an alternative formulation of sequent calculus : instead of sequents $\Gamma \vdash A$, one considers *deductions*

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

with a tree-like form. Every proof in sequent calculus induces a unique natural deduction, and conversely, every natural deduction comes from a sequent calculus proof, *but this ancestor is far from being unique* ! To explain the importance of this fact let us quote Prawitz (approximation of a private discussion, June '82)

<< J.Y.G. : I prefer sequent calculus which is more synthetic...

D.P. : maybe you are right. But sequent calculus is just a system of *derived* rules about proofs, whereas natural deduction tries to represent the proofs themselves as primitive objects. >>

This point is very central : sequent calculus is *the* synthetic way of manipulating those mysterious hidden objects (that we call proofs, programs, and we would like to see as actions). But natural deduction has been the first serious attempt to find out what these objects could be. To give an analogy : we can manipulate synthetic units like "tuner", "amplifier", "loudspeaker", and plug them together when certain matchings are fulfilled. Now, the hifi unit we thus obtain works by itself, without any reference to our decomposition into several units that was so essential to us : the resulting object is a complex mixture of transistors, diods etc., in which, by the way, other relevant synthetic units could have been individualized. In other terms : to build something (a proof, a program) we must go step-by-step and produce bigger and bigger synthetic configurations. But the object produced should not remember our particular step by step decomposition, which is purely taxonomic. Sequent calculus is presumably the best possible taxonomic system for actions. But an action may come from several descriptions, typically when

transposition of rules

occurs. The situation becomes dramatic with the *Hauptsatz*, where 90% of one's energy is spent on bureaucratic problems of transposing rules. Let us give an example : when we meet a configuration

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma', A}\,R \qquad \dfrac{\vdash A^{\perp}, \Delta'}{\vdash A^{\perp}, \Delta'}\,S}{\vdash \Gamma', \Delta'}\,\text{cut}$$

there is no natural way to eliminate this cut, since the unspecified rules (R) and (S) do not act on A or $A^{\perp}$; then the idea is to forward the cut upwards :

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\,\text{cut}}{\vdash \Gamma', \Delta}\,R}{\vdash \Gamma', \Delta'}\,S$$

But, in doing so, we have decided that rule (R) should now be rewritten *before* rule (S), whereas the other choice would have been legitimate too. Hence, from a symetrical problem, we are led to an asymetric solution : the taxonomical devices that force us to write (R) before (S) or (S) before (R) are not more respectable than the alphabetical order in the dictionary. One should try to get rid of them, or at least, ensure that their effect is limited. What natural deduction achieves is to identify intuitionistic sequent calculus proofs that are the same up to order of rules. (In classical logic, if we start with two proofs which are just variants w.r.t. transposition of rules, then the peculiarities of contraction on both sides are such that eventually the *Hauptsatz* will lead to two cut-free proofs which are not even variants.)

IV. 3 limitations of natural deduction

Natural deduction, which succeeds in identifying a terrific number of inversion-related sequent calculus proofs, is not free from serious defects :

α) Natural deduction is only satisfactory for $\Rightarrow$, $\forall$, $\wedge$; the connectives $\vee$

and ∃ receive a very *ad hoc* treatment : the elimination rule for "∨" is

$$\begin{array}{ccc} & \cancel{A} & \cancel{B} \\ & \vdots & \vdots \\ A \vee B & C & C \\ \hline & C & \end{array}$$

with the presence of an extraneous formula C. Then the problems of commutation of rules (i.e. changing the extraneous C) become prominent, and the solution of the literature (so-called "commutative conversions") is just *bricolage*.

β) In fact, there is a hidden taxonomy in natural deduction : in deduction

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

one distinguishes *one* conclusion, and *several* hypotheses. Since the conclusion is always unambiguous, there is no problem to determine which is the last rule used etc. The connectives ⇒, ∧, and the quantifier ∀ accept this taxonomy without any apparent problem. But disjunction would rather prefer a rule of the form

$$\frac{A \vee B}{A \quad B}$$

with two conclusions. But this would go against the taxonomic requirement "one conclusion at a time".

γ) In natural deduction, one distinguishes between *introductions* and *eliminations*. Our claim is that eliminations rules are just introductions (for a dual connective), but *written upside down*. For instance, the elimination rule for implication is written as

$$\frac{A \qquad A \Rightarrow B}{B}$$

and B is the official conclusion of the rule. But the *hidden* conclusion of the rule is A ⇒ B. This point should be familiar to specialists of natural deduction, where one introduces the "main hypothesis", which plays the role of the actual conclusion of a deduction. For instance, in order to cope with the lack of compositionality of "hexagons" in denotational semantics, this change of viewpoint becomes prominent, see e.g. [4] : in this paper, this shift of viewpoint yields the hexagon property for normal proofs (hence for all proofs, using normalization), whereas if one sticks to the usual taxonomy, one gets the impression that some compositionality is required.

δ) Natural deduction uses global rules, typically

$$\begin{array}{c} \cancel{A} \\ \vdots \\ B \\ \hline A \Rightarrow B \end{array}$$

which apply to whole deductions, in contrast to rules like the elimination rule for ⇒, which apply to formulas.

In the 70's, Rick Statman made an attempt [18] to study the geometry of natural deduction, and specially emphasized point δ), associating a "genus" to deductions. But since he had no way to restore symmetry, in order to cope with -say- point γ), the attempt eventually failed.

IV.4. proof-nets

Linear logic makes us hope that a final answer to the problem of inversion of rules might be found. This is because of the symmetrical nature of linear logic : the essential connective is now "$\multimap$", which conveys the purely implicative meaning of "$\Rightarrow$", putting aside the component "!", not of implicative nature. Hence, if one wants to study "$\Rightarrow$", we can study separately "$\multimap$" and "!". Here we shall concentrate on "$\multimap$", which in linear logic is defined from "$\parr$" by $A \multimap B =_d A^\perp \parr B$. The main idea will be to replace a natural deduction

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

by a *proof-net* with several conclusions, $\Gamma^\perp, A$. Everytime a formula will be turned upside down, the negation symbol $(-)^\perp$ will be needed.

α) when I write $\dfrac{\begin{array}{c}\cancel{A}\\ \vdots\\ B\end{array}}{A \Rightarrow B}$ in natural deduction, I can bend $\begin{array}{c}\cancel{A}\\ \vdots\\ B\end{array}$ so to get :

$$\frac{\overset{\cdots\cdots\cdots}{A^\perp \qquad\quad B}}{A^\perp \parr B}$$

this suggests the introduction of the (non global) binary rule :

$$\frac{C \qquad D}{C \parr D}$$

β) when I write $\dfrac{A \qquad A \Rightarrow B}{B}$ I can turn B and $A \Rightarrow B$ upside down ; this leads to the expression

$$\frac{A \qquad B^\perp}{A \otimes B^\perp}$$

which suggests another binary rule :

$$\frac{C \qquad D}{C \otimes D}$$

γ) minimal and maximal formulas in a deduction will be represented by means of configurations

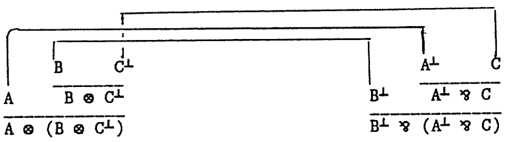
$$\frac{}{A \qquad\qquad A^\perp} \qquad \text{and} \qquad \frac{A \qquad\quad A^\perp}{\mathit{Cut}} \qquad \text{respectively.}$$

The first case (*axiom link*) represents an identity axiom, whereas the second case (*cut link*) is the cut-rule ; the expression *Cut*, which is not a formula, is there for inessential reasons.

This representation of proofs is the ultimate possible identification of proofs of sequent calculus. Typically the net below represents two algorithms :

on one side, the algorithm which takes a binary function f of type $A \multimap (B \multimap C)$ into a function g of type $B \multimap (A \multimap C)$ by interchanging the inputs

on the other side, the algorithm which takes a ternary sequence $b \otimes a \otimes c$ of type $B \otimes (A \otimes C^{\perp})$ into $a \otimes b \otimes c$ of type $A \otimes (B \otimes C^{\perp})$.



These two algorithms are physically the same because they correspond to the same pointer moves (represented by the axiom links) independently of the choice of the input side and of the output side. (By the way, observe that this proof-net is non-planar, i.e. wrong from the non-commutative standpoint.)

The main mathematical problem comes from the fact that we have expelled global (i.e. contextual) rules. Locally speaking the rules for $\otimes$ and $\wp$ are exactly the same. Hence one must find a global soundness criterion for the graphs written with such rules to be proof-nets, i.e. to represent (at least) one proof of sequent calculus. The answer is by means of the notion of a cyclic *trip*, which mimicks, in a formal way, the transportation of information during a cut-elimination process. For more details, one may consult [7] . However, this work is satisfactory only w.r.t. $(-)^{\perp}$, $\otimes$, $\wp$ (hence $\multimap$ as well). More recently, the notion of proof-net has been extended to quantifiers, see [9]. The solution found there could be the basis for a further extension to additives and exponentials, but what is known so far is only very partial. The weakening rules ($\perp$) and (w?) seem also to pose a very delicate problem, since they seem to contradict cyclicity.

IV.5. getting rid of syntax

This is the ultimate aim, which has been achieved only for multiplicatives, see [8], with essential improvements by Vincent Danos and Laurent Regnier [2]. In this case, the basic underlying structure turns out to be a permutation of pointers, and the notion of a cyclic permutation plays a prominent role (Danos and Regnier replaced cyclicity conditions by conditions involving connected acyclic graphs, i.e. trees). It seems that the general solution (not yet known, even for quantifiers) could be something like permutations of pointers, with variable addresses, so that some (simple form of) unification could be needed for the composition of moves : yet another connection with logic programming !

When one gets rid of syntax, an immediate question is

what is a type ?

Types have always been part of some rigid syntactic (i.e. taxonomic) discipline, while we would prefer a general answer, not depending on the choice of a particular system. Here we have to remember the familiar analogy between

"π is a proof of A" and
"π is a program enjoying specification A"

i.e. types are *specifications*. Specifications can be seen as *plugging instructions*. For instance one can plug something of type A with something of type $A \Rightarrow B$ and get something of type B. This is a particular case of the general paradigm of *plugging of complementary specifications*, which is the meaning of the cut-rule.

But let's assume that our program has been pushed to the end (!), so that we are without syntax. In the geometrical structure β representing a program π, let us individualize two arbitrary complementary parts, β' and β'', which communicate via a common border $\partial\beta' = \partial\beta''$. Does it make sense to "type" β' and β'' in such a way that these "typings" will ensure that, once plugged via their common border, β' and β'' yield a sound action ? The answer to this question is only known in the multiplicative case, and still open in the other cases, even for quantifiers. The short description below is taken from [8] (see also [2]), and we shall assume a complete familiarity with proof-nets :

any switching S of β can uniquely be decomposed as a disjoint sum $S'+S''$ of independent switchings of β' and β''. Using S', we get a certain permutation $\sigma_{S'}$ of $\partial\beta'$, and similarly, S'' yields a permutation $\tau_{S''}$ of $\partial\beta''$ (= $\partial\beta'$). The condition for β to be a proof-net is exactly the fact that $\sigma_{S'}\tau_{S''}$ is cyclic for all S', S''. Now, if we introduce

the notation $\sigma \perp \tau$
to say that σ and τ are two permutations defined on the same set, and that $\sigma\tau$ is cyclic

the sets of permutations
$$\Sigma(\beta') = \{\sigma_{S'}; S' \text{ switching of } \beta'\}$$
$$\Sigma(\beta'') = \{\tau_{S''}; S'' \text{ switching of } \beta''\}$$

then β is a proof-net if and only if :

$$\Sigma(\beta') \perp \Sigma(\beta'').$$

If we define the *principal types* of β' and β'' by :

$$\mathrm{pT}(\beta') = \Sigma(\beta')^{\perp\perp} \qquad \mathrm{pT}(\beta'') = \Sigma(\beta'')^{\perp\perp}$$

then β' and β'' can be plugged together exactly when

$$pT(\beta') \perp pT(\beta'')$$

In other terms, the principal type of an algorithm should be a collection of "border behaviours", permutations in the case just considered (more generally, something like partial isometries in a $\mathbb{C}*$-algebra ?). There is a notion of orthogonal border behaviours, cyclicity in the multiplicative case (more generally some kind of ergodicity ?). The principal type of an algorithm is not the set of all its border behaviours (which has a bad structure), but a larger set, its biorthogonal. (The reason is that if we take an algorithm of type $A\otimes B\otimes C$, we shall not get the same border behaviour whether we write it using $A\otimes(B\otimes C)$ or $(A\otimes B)\otimes C$, but this difference, due to taxonomy, is swallowed by the use of biorthogonality.

Of course, it would be a nonsense to try to compute principal types, as defined above. But plugging requires orthogonality of the principal types, and not at all that each of them is the orthogonal of the other, so that there is room in between. In particular, preset systems of typing can be seen as convenient ways to get and manipulate majorizations of principal types. The practical way of showing that β' is pluggable with β'' is therefore to find convenient majorizations

$$pT(\beta') \subset B' \qquad pT(\beta'') \subset B'' \qquad \text{with } B' \perp B''.$$

By the way, the types we attribute to algorithms in the multiplicative fragment are always majorizations, and are seldom optimal.

There is still the problem of how to interpret cut-elimination. If we could see an action as something as a partial isometry p in -say- a $\mathbb{C}*$-algebra, then an action is performed (cut-free) when $p^2 = 0$, i.e. when its domain X and its codomain Y are orthogonal subspaces : $XY = 0$. For a general action p (corresponding to the idea of a proof with cuts), it would no longer be true that $XY = 0$, but X would commute with Y, so that it would make sense to speak of the projectors (subspaces) $X' = X - XY$ and $Y' = Y - XY$. The syntactic operation of cut-elimination should be geometrically translated as a construction leading from p, with domain X and codomain Y, to p', with domain X' and codomain Y', the idea being to start from X' and to iterate p the number of times necessary to exit through Y'. For instance the process of cut-elimination in the multiplicative case can perfectly be interpreted in this way. However the consideration of finite dimensional spaces would be enough in that case ; the introduction of more abstract spaces seems to be necessary in order to represent irreversible processes like erasing, or simply to make room for duplications when we shall interpret contraction. An attempted treatment of actions as partial isometries in Hilbert space was started in Spring '87. It turned out that this approach could not replace down to earth syntactic considerations, in other terms that not enough material had been accumulated for a decent conceptualization ; but we nevertheless still believe that Gentzen's Hauptsatz should eventually be reformulated using the language of functional analysis !

IV.6. time, space, communication

Linear logic is eventually about time, space, communication, but is not a temporal logic, or a kind of parallel language : such approaches try to develop preexisting conceptions about time, processes, etc.. In those matters, the general understanding is so low that one has good chances to produce systems whose aim is to *avoid* the study of their objects (remember the sentence of Clémenceau : <<Quand je veux enterrer un problème, je nomme une commission>>. Linear logic is not "la commission du temps" or "la commission de la communication". The main methodological commitment is to refuse any *a priori* intuition about these objects of study, and to assume that (at least part of) the temporal, the parallel features of computation are already in Gentzen's approach, but are simply hidden by taxonomy. We shall therefore search for the answers to these essential problems inside *refinements* of usual logic, and not in such and such *ad hoc* extensions.

Methodologically speaking we concentrate on the central issue of

unbracketing.

We are forced to program sequentially, by means of nested brackets. This bracketing is a particular temporality for a program : if we perform the operations in the order imposed by the brackets, then we shall eventually make it. However, there might be other temporalities which may for instance come from some unexpected property of the inputs, and which might be more interesting. The idea of removing taxonomy, i.e. *our* temporality, by means of something like proof-nets, is to give the maximum degree of freedom for the execution of a program. The extension of proof-nets to quantifiers yields some additional hints as to a possible nature of this temporality, namely the relative dependencies between variables in massive processes of unification. In general, one should see time as the partial order of causality ("I must compute this to get that"), the absence of causality being perhaps of spacial nature. Negation would therefore appear as the inversion of the sense of time, which is not an unpleasant idea. The main problem one is faced with is that we have at least three intuitions about time :

i) time is logic modulo the order of rules
ii) time is the cut-elimination process
iii) time is the contents of non-commutative linear logic

These three intuitions should be unified to some extent ; however, one of the immediate difficulties with ii) is that cut-elimination is, at it stands, an irreversible process (which is consistent with our current experience with time) whereas the logical rules of iii) are symmetric w.r.t. the exchange past/future. Moreover, technically speaking, there are problems to develop iii), namely to have simultaneously commutative (spacial) and non-commutative (temporal) features.

As to the symmetry of time, there is a new element coming in : in the study of quantifiers, the normalization process uses some global procedures, which seem to involve a global time. This is a computational nonsense (since the thing will anyway be implemented by a local procedure) which seems to come from the symmetry between past and future. If one drops this symmetry, i.e. if the implication

$$\forall xA \Rightarrow A[t/x]$$

is only kept as a retro-causality, but not as a causality :

$$A[t/x] \leftarrow \forall xA, \qquad \text{(but not} \quad \forall xA \rightarrow A[t/x] \text{)}$$

it seems possible to keep local procedures, but it is very difficult to find any milestone on which we could test such possible refinements of rules. But the idea that starts to make its way is that

logical rules should not be symmetric w.r.t. time.

A last word about communication : on the basis of the work so far done, the following conception of communication between systems seems to be reasonable :

processes communicate without understanding each other.

The idea should be that -at a very abstract level-, what processes share is a common border, but that their inner instructions have nothing in common. So when A receives a message from B, he can only perform global operations on it

-erasing, duplicating, sending back to B-depending on which gate of the common border he received it through. When a message is sent back to B, then B receives again his own stuff, that he can read, but through an unexpected gate etc. Massive iterations of such incomprehensions can perhaps mimick comprehension ; by the way this corresponds to the way scientists use the patience of their colleagues to improve their intuitions about their own work.

Finally, we must confess that we keep an eye on physics, especially quantum mechanics. It is not excluded that strange linear connectives like "⅋" could be useful to interpret some basic phenomena of physics...
But this is science-fiction.

REFERENCES

[1] Barr, M. :
**-autonomous categories*, Springer Lecture Notes 752.

[2] Danos, V. & Regnier, L. :
Multiplicatives bis, draft, université Paris VII, 1988.

[3] Dunn, J.M. :
Relevance logic and entailment, Handbook of philosphical logic, vol III, ed Gabbay & Guenthner, D.Reidel 1986.

[4] Freyd, P. & Girard, J.Y. & Scedrov, A. & Scott, P. :
Semantic parametricity in typed λ-calculus, proceedings of the Congress "Logic in Computer Science 1988", to be held in Edimburgh.

[5] Girard, J.Y. :
Une extension de l'interprétation fonctionnelle de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types, Proc. Second Scand. Log Symp., ed. Fenstad, North Holland 1971.

[6] Girard, J.Y. :
Proof-theory and Logical complexity, Bibliopolis, Napoli, 1987, ISBN 88-7088-123-7.

[7] Girard, J.Y. :
Linear Logic, Theoretical Computer Science 50:1, 1987.

[8] Girard, J.Y. :
Multiplicatives, Rendiconti del seminario matematico dell'università e politecnico di Torino, special issue on logic and computer science, 1988

[9] Girard, J.Y. :
Quantifiers in linear logic, to appear in the Proceedings of the SILFS conference, held in Cesena, January 1987.

[10] Girard, J.Y. :
Typed λ-calculus, in preparation for Cambridge Tracts in Theoretical Computer Science.

[11] Girard, J.Y. & Lafont, Y. :
Linear logic and lazy computation, Proceedings of TAPSOFT '87, Pisa. SLNCS 250.

[12] Howard, W.A. :
The formulae-as-types notion of construction, in Curry Volume, eds Hindley & Seldin, Academic Press, London 1980

[13] Ketonen, J. & Weyhrauch, R. :
A decidable fragment of predicate calculus, Theoretical Computer Science 32:3, 1984.

[14] Lambek, J. :
The mathematics of sentence structure, Am. Math. Monthly 65, 1958.

[15] Lambek, J. & Scott, P. :
Introduction to higher order categorical logic, Cambridge University Press, Cambridge 1986

[16] Prawitz, D. :
Natural Deduction, Almqvist & Wiksell, Stockholm 1965.

[17] Scott, D. :
Domains for denotational semantics, Proceedings of ICALP '82, SLNCS 140.

[18] Statman, R. :
Structural complexity of proofs, Ph. D., Stanford, 1975.

Équipe de Logique, UA 753 du CNRS
Mathématiques, t 45-55, 5° Étage
2 Place Jussieu, 75251 Paris Cedex 05