

B-2 – Description du projet et résultats attendus (8 pages maximum en Arial 11, simple interligne)

L'originalité et le caractère ambitieux du projet devront être explicités. L'interdisciplinarité et l'ouverture à diverses collaborations seront à justifier en accord avec l'orientation du projet. La capacité de ou des équipes « porteuse(s) » devra être attestée par la qualification et les productions scientifiques antérieures de leurs membres. Leurs rôles dans les différentes phases du projet devront être précisés et la valeur ajoutée des collaborations entre les différentes équipes sera argumentée. On décrira le déroulement prévisionnel et les diverses phases intermédiaires ainsi que les méthodologies employées. Les moyens demandés devront être en accord avec les objectifs scientifiques du projet.

Uniquement dans le domaine des sciences humaines et sociales, les projets de recherche peuvent impliquer la production de données statistiques. Dans ce cas l'accès au financement de l'ANR implique l'obligation de déposer ces données, documentées, dans un centre d'archivage et de diffusion auprès des chercheurs, et de les mettre à disposition de la communauté scientifique (éventuellement au terme d'un embargo de durée déterminée).

1 Introduction

The *Curry-Howard (C-H) correspondence* offers a major conceptual bridge between mathematical logic and computer science, and a fruitful paradigm in the developments of both proof theory and computer science. The correspondence has been originally discovered in intuitionistic logic, where it revealed the deeply logical nature of the *lambda-calculus*, one of the main definitions of computable functions. The correspondence enables to see logical formulae as types, and conversely, to think of types as logical specifications for the behavior of programs. This theoretical point of view has brought very concrete results in the 1980s and 1990s, leading to the application of logical tools to the programming language technology, and the development of programming languages like CAML, and of proof assistants like Coq.

The C-H correspondence has been extended to larger fragments of logic, and in particular to classical logic, this providing a highly operational viewpoint on classical reasoning principles, such as the Law of Peirce or the Excluded Middle. The C-H correspondence is also at the origin of Linear Logic, one of the main conceptual and technical tools for understanding the operational content of proofs.

Our goal, as explained in B-1, is to extend the C-H correspondence to a much larger domain, including process algebras which model concurrent and mobile processes. From a success in this attempt, we expect new mathematical tools for dealing with concurrent programs (and more specifically for proving their correctness) and new theoretic guidelines for designing better behaved process algebras. Recent progresses performed at the interface between Linear Logic and Concurrency Theory allow us to be optimistic in this notoriously ambitious task.

We first describe the scientific context of our proposal.

2 General scientific context

After the work of Gentzen in the 1930's, it became clear that the process of *cut-elimination* is at the heart of the most important question in traditional proof-theory, namely, the problem of relative consistency (absolute consistency being out of reach in general by Gödel incompleteness theorems). For instance, system T and system F are typed lambda-calculi whose strong normalization – that is, termination of cut-elimination – guarantees the consistency of Peano and second order arithmetics respectively. Therefore these normalization results can only be proved in strictly stronger logical systems: for instance, the proof of strong normalization of system T cannot be done within Peano arithmetics, whereas it can be performed in second order arithmetics. This kind of considerations led to many developments in proof theory, allowing for a classification of logical systems according to the complexity of their cut-elimination process.

Cut-elimination as computation. But the cut-elimination process has the much more general significance of a *universal computational paradigm*; this fact has been first observed in propositional intuitionistic logic. The proofs written in this system can be seen as lambda-terms, that is, as terms of the

lambda-calculus, a formalism introduced in the 1930's for modelling general computations – indeed, together with Turing machines and the formalism of recursive functions, the lambda-calculus, with beta-reduction as a notion of computation, provides one of the mathematical definitions for the concept of « computable function », all the corresponding definitions being extensionally equivalent (Church thesis). This correspondence is not only a formal identification of syntactical objects (proofs and lambda-terms), but also an exact identification between two computational processes: cut-elimination and beta-reduction.

This simple observation had then a lot of important consequences on the developments of both proof theory and theoretical computer science, and initiated a process of « exchange of technologies » between the two fields. The most spectacular example of this phenomenon is probably *denotational semantics*. Initially conceived by Dana Scott for providing a mathematical interpretation of programs and lambda-terms, it became an essential tool in proof-theory, providing mathematical invariants of cut-elimination and suggesting new logical systems: the best example is of course *Linear Logic* which takes its origins in the coherence space denotational model of system F built by Jean-Yves Girard, see below.

Extension of Curry-Howard to classical logic. Initially restricted to intuitionistic logic, the C-H correspondence has then been extended to classical logic. The key observation came from computer science, where it has been observed that the call-cc (*call with current continuation*) instruction of the *Scheme* programming language can safely be typed using the *Law of Peirce*, a formula provable in classical propositional calculus but not in intuitionistic propositional calculus. Such instructions are very important in the practice of programming because they allow for some « non local » interactions between the various pieces of a program, as for instance when one uses *exceptions*. It was also observed that the standard method used by computer scientists for translating such instructions as lambda-terms, known as *continuation-passing style* (CPS) translations exactly corresponds to a method invented much earlier by Gödel for translating classical logic in intuitionistic logic: the *not-not translation*. Cleaned up, these ideas led later to the *lambda-mu calculus* [Parigot1], a well-behaved classical extension of the lambda-calculus.

Realizability. Typing can be considered as a particular case of a more general relation between a term (lambda-term or lambda-mu term) and a formula: realizability. This concept, defined by induction on logical formulae, is also known in Computer Science under the name of « logical relations ». A term realizes a formula when it satisfies the formula, considered as a program « specification »... however understanding in what sense a formula is a specification can be quite difficult (unless the formula has a particular form; cf. the work of Krivine on realizability in Set Theory). Danos and Krivine [Danos,Krivine] have extended this approach to languages having parallel features (lambda-mu calculus with MIX) for realizing particular classical formulae with coroutines, and, very much in the same spirit, Beffara developed in his PhD thesis a realizability of linear logic formulae by means of processes from the pi-calculus (or similar formalisms). However, as we shall see, in this kind of approach, the processes which are realisers tend to have very deterministic behaviours, quite remote from the situations one usually wants to describe using process algebras.

Linear logic. The crucial discovery of linear logic by Jean-Yves Girard in 1985 [Girard1] radically changed the landscape of proof-theory and also of the abstract theory of programming languages in theoretical computer science (it is interesting to notice here that the article « Linear Logic » published in the journal *Theoretical Computer Science* in 1987 has won the award of the most often quoted article of this very well established journal). In a typically Curry-Howard oriented perspective, Girard started from the coherence space denotational model of system F. He observed in this model some striking symmetries and a decomposition of the intuitionistic implication using two new connectives: a binary connective called *linear implication*, and a unary connective (or modality) called the *exponential*. He developed then the corresponding logic, *Linear Logic (LL)*, where the structural rules (weakening and contraction) of intuitionistic or classical logic got, for the first time, a full logical status, with associated exponential connectives. What differentiates linear logic from the previous logical systems is that it is a *resource sensitive* logic; exponential connectives are concerned with structural rules (weakening, that is – through the C-H correspondence – resource discarding, and contraction, that is, resource duplication). For that reason, the concept of *linearity* plays a crucial role: a proof is linear if it uses its hypothesis exactly once, no duplication and no discarding can occur on its argument.

Proof-nets. Apart from shedding a new light on the already known Curry-Howard interpretations of intuitionistic and classical logic, linear logic also gave rise to more canonical representations of proofs: *proof nets* (Girard) [Girard1] and *interaction nets* (Lafont, IML) [Lafont1]. These settings allow asynchronous reduction of cuts – a clearly necessary, but not sufficient, step towards an extension of C-H to concurrent computations. Contrarily to sequent calculus or natural deduction (lambda-calculus) proofs, which are trees, proof-nets are graphs which must satisfy a global, topological, *correctness criterion* [Girard1,Danos-Regnier1] in order to correspond to a « real » proof.

Geometry of interaction, games and denotational semantics. Also, linear logic widened the world of proof semantics. It recasted the Lawvere « cartesian closed » viewpoint on intuitionistic logic (a denotational model of intuitionistic logic is essentially a cartesian closed category) in a more algebraic setting, where proofs are basically interpreted in a *symmetric monoidal closed category*, something which looks like a category of vector spaces, the (involutive) negation of linear logic corresponding to duality [Bierman1]. The connectives of linear logic correspond then to standard constructions in linear algebra (tensor product, its dual which is not necessarily identical to the tensor product – as would be the case in finite dimensional vector spaces, direct product and co-product). One retrieves this kind of categorical structures in all denotational models of linear logic, with an additional structure corresponding to the exponential modality: an endofunctor on the linear category which has a structure of comonad satisfying a number of conditions.

Beyond this kind of purely static models, linear logic allowed to introduce a new kind of interpretation of proofs and of cut-elimination: the *geometry of interaction* (GoI) [Girard3], where proofs are interpreted as operators acting on some space of states. Concrete versions of the GoI allowed for a refined understanding of beta-reduction, and in particular of the optimal reduction problem as addressed by Lamping [Danos-Regnier2]. GoI is now widely understood as a computation of traces in certain traced monoidal categories, but Girard is also developing an alternative viewpoint, based on the theory of Von Neuman algebras, which might establish a deep connection between linear logic and quantum mechanics in a near future [Girard4].

The 1990's have also seen the birth (or rather, rebirth) of *game semantics*, a dynamic semantics of proofs and programs. The originating ideas actually date back to an attempt of Gentzen to develop in the 1930's an interactive account of cut-elimination, where formulas are interpreted as games, and proofs are interpreted as winning strategies. This very natural idea of an interactive interpretation of computation arose then independently in various contexts: at the end of his career, Kleene developed similar ideas for establishing the theory of higher order computable functionals on clean mathematical grounds. Independently and more or less in the same period, Berry and Curien (PPS) [Berry-Curien1] developed the theory of *sequential algorithms* which interprets functional programs of all simple types as output-driven interactive devices. Game semantics unifies these various traditions, by interpreting a type (or a formula) as a two-player game, where a program (called Proponent) interacts against its environment (called Opponent) according to a deterministic strategy satisfying certain additional requirements (innocence, history-freeness, visibility, positionality, etc.) depending on the underlying programming language. Linear logic played an important role in the recent developments of game semantics because the Player/Opponent symmetry corresponds precisely to linear negation [Blass1]. One main achievement of game semantics has been the construction of *fully abstract* (or more precisely: denotationally complete) models -- that is, models where all the elements are definable in the language [Abramsky-Jagadeesan-Malacaria,Hyland-Ong]. These theoretical developments have led to very concrete applications, and in particular to model checking tools for validating programs, based on a game-theoretic approach to abstract interpretation (cf. the work of Dan Ghica at the University of Birmingham [Ghica1]).

Game models are also strongly connected to the GoI [Abramsky-Jagadeesan,Baillo], and can be seen as giving an abstract description of very concrete evaluation devices of lambda-terms [Danos-Herbelin-Regnier1] like the Krivine Machine [Krivine3].

Concurrency and process calculi. The 1980's have also been characterized by an explosion of the theory of process calculi, motivated by the increasing need of theoretical and practical tools for validating

concurrent systems, mobile software, communication protocols. Petri Nets, Mazurkiewicz traces, and then more syntactic formalisms proposed by Robin Milner (called *process algebras*: CCS, and then the *pi-calculus* [Sangiorgi-Walker] and many variants of this calculus defined by various authors) have been developed for modeling parallel computations.

Process algebras have found applications: in security, with the formalization of cryptographic protocols made possible by the spi-calculus [AbadiGordon]) and in formal biology, especially for expressing interaction chains between proteins, notably. One should mention here the work of Luca Cardelli (see his web page [Cardelli1]).

On the theoretical side, many results have been proved concerning the expressive power of process calculi, the various notions of simulation between processes and the codings of the lambda-calculus in the pi-calculus. In concurrency theory indeed, one is used to consider processes as organized in a *transition system*, a labeled graph representing all possible evolutions of the system, depending on its possible interaction with its environment (and not on its *internal* dynamic capabilities). Then two processes are bisimilar if, essentially, they have the same potentiality of evolution in their transition system (the precise mathematical definition depends on what is precisely observable in the evolution). This is a very interesting approach as it hides a lot of bureaucratic aspects of processes (the syntaxes of process algebras have usually a lot of defects and unwanted features), but the price to pay is the high complexity of the concept of bisimulation and, often, its poor properties with respect to the syntax of processes itself: quite often, for instance, bisimulations are not congruences, and one has to close them under arbitrary contexts. And then it can become quite difficult to prove that two processes are bisimilar. The theory of bisimulation in process algebras for mobility has been studied in depth by D. Sangiorgi, notably by classifying equivalences and inventing new proof techniques. In particular, the study of calculi that feature higher-order communication gives hope that some of the techniques that have been developed might be applicable in a sequential setting, and in particular to reason on the lambda calculus.

Typing systems have been proposed to check that the interactions between concurrent processes obey a certain discipline. This makes it possible to guarantee properties like the good usage of channels, or, in some cases, the absence of deadlocks or even normalization (cf. in particular the works by Sangiorgi and by Honda). Also *causal* semantics have been developed for process calculi among which the most satisfactory approach is probably the presheaf model designed by Glynn Winskel (Cambridge) [Winskel1]. This approach generalizes the classical *trace* interpretation [Mazurkiewicz], and also the more expressive *event structure* interpretation of processes: in the trace interpretation, a process is represented as the set of all possible sequences of events for the considered process, these sequences being of course organized as a tree; in the event structure approach, the events are organized in a more general poset (directed acyclic graph) in which both causality and incompatibility of events are represented. Extending this kind of approach to presheaves (to be understood as « set valued sets ») had considerable benefits, allowing in particular to describe the concept of bisimulation in terms of *open maps*, cf. the work of Winskel and Joyal [JoyalWinskel].

Fundamental analogies. As we said in B-1, there are many features of the theory of process algebras which have strong, though not well formalized, analogues in proof theory and lambda-calculus; let us recall them with a bit more details.

1. Transition systems are similar to rewriting systems, with the difference that they are labeled by names corresponding to interactions with a potential environment. So (apart for the silent *tau* transitions) a transition from a process is not associated to a redex inside the process, but with a potential reaction of this process to a signal of its environment.
2. Bisimulations are similar to the equivalences induced by these rewriting systems. *Barbed bisimulation* [MilnerSangiorgi] is closer to an «observational» equivalence, a quite standard concept in lambda-calculus. This viewpoint has been recently generalized by Rathke, Sassone and Sobocinski using ideas coming from realisability and LL ([RathkeSassoneSobocinski]).
3. Traces (and their generalization: event structures, presheaves) are similar to games and to designs in *ludics* (a game-like logical system designed recently by Girard), or Curien and Faggian's L-nets.

4. The connection between processes and spatial formulas is similar to realisability, except that the spatial logics used for reasoning about processes tend to have non-standard connectives whose logical meaning is not entirely clear, and a proof theory (cut elimination etc) which remains to explore.
5. Most process typing systems bear similarities with lambda-calculus typing systems, and allow typically to avoid "execution errors" due to misuse of communication channels. But, according to the C-H paradigm, these typing systems should be related to the logical systems of (4), and this is not the case yet.
6. Replicable processes are similar to boxes in linear logic: this analogy is extremely clear, and LL was probably a source of inspiration for Milner when designing the pi-calculus. However, replicable processes can have more complex interactions with their environment than LL boxes.
7. Locality in processes is reminiscent of both linear logic boxes and of locality in ludics.

In spite of these striking analogies, until quite recently as we shall see, nothing similar to a convincing Curry-Howard interpretation of fully concurrent and non-deterministic process calculi was available.

Our project will be an attempt at giving a more concrete and mathematically satisfying content to these fundamental analogies, with, as a result, such a C-H correspondence for processes.

3 Main topic, methodology, tasks and expected achievements and benefits.

We have good reasons to be optimistic as to the realization of this notoriously ambitious task, because of recent theoretical progresses performed in at least three converging directions: differential interaction nets (DIN), causal semantics of proofs and processes, and realizability of logic in process algebras.

Differential interaction nets (DIN). After their work on the differential lambda-calculus [Ehrhard-Regnier1], Thomas Ehrhard (IML and PPS) and Laurent Regnier (IML) have proposed an extension of ordinary proof-nets of linear logic, corresponding to the differential lambda-calculus (these nets are presented using the more convenient formalism of *interaction net* introduced by Yves Lafont) [Ehrhard-Regnier2]. This setting fundamentally extends the exponentials of linear logic by adding new rules, dual to the ordinary weakening, dereliction and contraction rules. In that way, and without breaking the strong typing discipline of linear logic, a tamed non-determinism appears, and also, thanks to the contraction/co-contraction interaction, a many-to-many interaction mechanism is made available, very typical of process calculi. In spite of this, the main fundamental properties of cut elimination (confluence, strong normalization) are preserved in this extended setting; in particular, confluence does not contradict non-determinism because the result of a non-deterministic reduction is a sum of nets. One should notice that DiLL radically modifies the viewpoint on the exponentials of linear logic, and this explains the connection between DiLL and processes: *exponentials are connectives dealing with communication and not only with duplication*. For instance, the box-free fragment of DiLL has exponentials (this is not the case of LL since the only way of introducing the « bang » connective is the box construction) but no real duplication capabilities. In this system, the effect of contraction is not of duplicating structures, but of splitting them into pieces (in many different ways, whence the sums appearing in the reduction).

One of the nice features of differential interaction nets (or more generally of DiLL) is that they admit a very appealing mathematical interpretation, based on elementary operations of the ordinary differential calculus (the non-determinism sum being interpreted as the ordinary mathematical sum in vector spaces): for instance, the *co-dereliction* logical rule of DiLL corresponds to taking the derivative of a function at 0. In the corresponding denotational models of DiLL, (intuitionistic) proofs can be seen as *analytic functions* and therefore are subject to differential operations [Ehrhard1]. The two basic « non-deterministic » reduction rules are dual of each other and correspond to the *product rule* in differential calculus and to the fact that linear functions commute to sums, respectively.

Starting from the observation that the typing system considered by Berger, Honda and Yoshida in [BergerHondaYoshida1] can be interpreted in polarized linear logic, Olivier Laurent and Kohei Honda designed a faithful translation of a relevant fragment of the pi-calculus in an extension of LL proof nets which can naturally be seen as a subsystem of DiLL interaction nets with boxes. Thomas Ehrhard and

Olivier Laurent observed that apparently this translation can be made modular by using more systematically the structures of DiLL: suitable combinations of contraction and co-contraction cells of DiLL allow for building structures which model *communication areas* associated with names and to which processes can be connected for interacting with each other. This seems to be an adequate tool for modelling, in a modular way, the parallel composition construction of process calculi. If this is the case and if the translation behaves well wrt. reduction, this would mean that there is a *modular* C-H correspondence between the pi-calculus and DiLL, a logical system. Also, the use of full DiLL should allow to interpret a larger fragment of the pi-calculus.

Encouraging is the fact that we are now able to interpret a finitary pi-calculus (a polyadic pi-calculus without replication and without sums) in DiLL [EhrhardLaurent1]. This translation is based on the following ideas:

- Parallel composition and name restriction are interpreted with communication areas;
- Input and output prefixes are interpreted with codereliction and dereliction, and with multiplicatives for combining the object name into tuples;
- Sequentiality (the fact that a prefix which stands behind another one cannot react) is implemented thanks to multiplicatives as well.

This latter point is particularly interesting since purely asynchronous process calculi, such as the calculus of solos, are more easily translated to DiLL, and it was not clear that such a faithful translation would be possible for the pi-calculus as well. Our encoding of the pi-calculus sequentiality is inspired from a translation of the pi-calculus in solos [LaneveParrowVictor]. The main feature of this translation is that it provides a *toolbox* for representing concurrent processes in DiLL, a set of elementary nets which can be combined for representing complex concurrent behaviours.

Causal semantics of proofs and processes. Describing the causal or asynchronous semantics of concurrent computations is an old tradition in Concurrency Theory, which culminated in the 1980's, when Glynn Winskel (Cambridge University) designed a truly concurrent semantics of CCS, based on event structures. The model refines the *interleaving* semantics of CCS designed previously by Robin Milner (Cambridge University) using synchronization trees.

This truly concurrent semantics of CCS was recently extended by Daniele Varacca (PPS) and Nobuko Yoshida (Imperial College) to a linearly typed variant of the pi-calculus. The semantic construction relies on a typing system for event structures, which enforces a *confusion-freeness* property, stating that every non-deterministic choice is localised in the event structure, and does not depend on the scheduling of independent components. Composition between typed event structures is shown to preserve the linear types, and thus to induce a confusion-free event structure. This fundamental property leads to the first model of (a well-behaved fragment of) the pi-calculus based on event structures.

We would like to connect and possibly unify this causal semantics of the pi-calculus to the translation in differential interaction nets mentioned above, using game semantics. Game semantics delivers indeed a *trace semantics* of programming languages, where a program (or a proof) is interpreted as the interactive strategy generated by its symbolic traces – a trace here means a sequence of program transitions, called *moves*. One main achievement of game semantics has been the definition in the 1990's of a fully abstract model of the functional language PCF by Martin Hyland (Cambridge University) and Luke Ong (Oxford University). The causal nature of this model has been later uncovered by Paul-André Melliès (PPS) who reformulated it using *asynchronous games* played on event structures. Starting from this work, Melliès has recently designed a game model of *full* propositional linear logic. This construction solves an old and notoriously difficult problem, called *Blass problem* by Samson Abramsky: all the previous traditional game models are indeed limited to (intuitionistic or polarized) fragments of linear logic. We would like to extend this game semantics to differential linear logic, starting from a categorical analysis of its two main features: sums and differentiation. The resulting model should revisit the game model of non-determinism designed by Vincent Danos and Russ Harmer [Danos-Harmer].

The theory of *concurrent ludics* and *L-nets* developed by Pierre-Louis Curien and Claudia Faggian (PPS) will play a key rôle in the convergence of our various models. It delivers indeed a framework at the interface between game semantics, event structures, and a concurrent and non-deterministic variant of proof-nets (*L-nets*). Moreover, the truly concurrent model of the pi-calculus designed by Varacca and

Yoshida was recently connected to L-nets by Faggian and her student Mauro Piccolo.

Realizability of logic in process algebras. In concurrency, logical systems have been developed in order to express operational properties of processes (for instance: can a process be divided in various subprocesses?). These logics have modalities dealing with time and have connectives and modalities oriented towards these expressivity requirements; therefore they are quite close to the actual syntactic primitives used for building processes. Such logics allow to express for instance that a computation is made of two activities taking place at different sites. They induce logical equivalences on processes (two processes being equivalent if they satisfy the same formulae). Daniel Hirschhoff and Tom Hirschowitz are comparing these logically defined relation with usual behavioural equivalence, raising the question of where and when behavioural is sensitive to locality phenomena. Even in a "simple" setting like the pi-calculus, where a parallel composition of processes is interpreted as two activities taking place at different sites, this question is interesting, and has connections with the question of context closure of bisimilarity.

Just as in realizability interpretations of (intuitionistic, classical or linear) logic, the semantics of a formula of such logics is a set of « programs » (here processes), but on the other hand, these logics seem too specifically designed for their purpose and therefore lack standard purely logical properties such as well behaved proof systems enjoying a normalizing and confluent proof system. More precisely, proofs of formulae should be of a nature similar to that of their realizers: in intuitionistic or classical logic, realizers are lambda-terms, possibly extended with additional primitives, among which certain well-formed terms are proofs; the fundamental property states that proofs of a formula are realizers of that formula, but the converse is false in general: in Krivine's programme, some of the axioms of Set Theory have quite interesting realizers. Recent results obtained on realizability of Linear Logic formulae by concurrent processes (in particular, the work of Beffara [Beffara]) are encouraging, but, due maybe to the interpretation of the tensor product in terms of parallel composition, it seems that in this kind of approach, realizers tend to be deterministic processes, closer to lambda-terms than to true concurrent processes. The shift suggested by the interpretation of processes as DIN's opens new perspectives as logical rules are intrinsically associated to the basic cells of DIN's. But of course the connection between the logical formulae and their « spatio-temporal » meaning (in the sense that the fact for a DIN to realize a formula should have such a meaning) will be much less obvious than in the settings developed so far.

We want to re-address the fundamentals of these space and/or time oriented logics of processes in view of the progresses accomplished in Linear Logic, and especially in view of DiLL. Indeed, DIN's seem to provide both a language for representing concurrent processes and also contain a « logically correct » part (the nets which satisfy the correctness criterion) which represents the proofs of a logical system (DiLL, which contains LL, and hence intuitionistic and classical logics as subsystems). So DIN's seems to meet the requirements of a well behaved realizability theory in a C-H compatible setting, extending moreover the already known lambda-calculus oriented realizability theories. This latter point is essential: it is much more interesting, and also challenging, to extend an already existing and theoretically developed framework, such as standard realizability theory - which is itself related to many standard constructions such as Cohen's forcing - than reinventing something completely new « from scratch ». From this viewpoint, one of the most important features of DIN's is that they are an *extension* of LL, a completely standard logical system, with clear connections to intuitionistic and classical logic.

Crucial choices will have to be made, in particular whether spatial or temporal modalities should be added to linear logic in order to get a sufficiently expressive system - experience shows that one has to be extremely cautious with this kind of extensions since we know from realizability in Set Theory that operational properties expressed by formulae can be quite far from their usual (model-theoretic) meaning -. The extension of LL by such modalities or constructions will have to fulfill a number of constraints: a well-behaved cut-elimination should still be available, as well as a natural denotational semantics, extending the standard relational semantics of LL. This might be a good opportunity of considering seriously first-order predicate calculus version of LL, and also to exploit the *locative* aspects of *ludics*. On this aspect, the experience of Curien and Faggian (PPS) and also of members of the Logique de la Programmation group of IML (Donnadieu and Quatrini, not officially taking part to the present proposal) will be quite helpful. In understanding the logic of processes in our new setting, we shall in particular benefit from the experience of the LIP group (Hirschowitz and al.), which has a deep

practice of this topic, and has in particular explored the equivalences induced by spatial logical systems on processes (two processes being equivalent when they satisfy the same formulae).

Objectives and tasks. To summarize, we shall describe a number of tasks that we plan to accomplish in our project. They are related to the various contact points between process theory and proof theory developed above. It would be meaningless to give a timetable for the accomplishment of these objectives apart for TASK 5, which will need progresses accomplished in the other tasks, and therefore will not be addressed before the second year of the project. These tasks are of course deeply related with each other and the splitting between them is somewhat artificial.

TASK 1: transition systems, rewriting and equivalences of processes and DIN's. Understand the connection between these two approaches to the dynamics of processes, the first one being used in the standard setting of process algebras and the second one seeming more appropriate to the description of DIN's. Understand the connection between the associated equivalences of processes, and with the equivalence induced on DIN's by their denotational semantics. **Researchers in charge of this task:** Ehrhard, Laurent (PPS), Hirschhoff, Hirschowitz, Sangiorgi, Hyvernat (LIP), Regnier (IML).

TASK 2: typing and realizability of processes. Use systematically LL for typing processes, design first order, second order and recursive type systems based on LL, using translation of processes in DIN's (which admit such typings). Explore the normalization properties of typed processes and nets, introducing also correctness (acyclicity) criteria on nets; what are the processes whose translation in DIN satisfy these criteria? Analyze the connection between processes (nets) and formulae in terms of realizability and explore the expressive power of LL in terms of space and time. Consider extending LL for increasing this expressive power. **Researchers in charge of this task:** Ehrhard, Laurent, Beffara, Mazza (PPS), Hirschhoff, Hirschowitz, Sangiorgi (LIP), Regnier (IML).

TASK 3: causal semantics of proofs and processes, game semantics and the GoI. Develop the causal semantics of the pi-calculus, compare to concurrent ludics and L-nets, define an asynchronous game model of differential linear logic, compose this game semantics to the translation of the pi-calculus in DIN's, and compare the resulting causal semantics to the existing semantics of the pi-calculus. **Researchers in charge of this task:** Curien, Faggian, Melliès, Varacca (PPS), Santocanale, Morin (LIP).

TASK 4: location, replication and boxes. Consider process algebras with locality constructions, such as *ambient calculi*. Try to understand if such extensions have their counterpart in DiLL, using possibly locative formulae (as in ludics), or *jumps* (again, as in ludics, extended by Faggian and Maurel to the L-net setting), or even *boxes* (as in proof-nets and more generally, string diagrams). Study the use of exponential boxes of LL for implementing replicable pi-calculus processes in DIN's: how far are we from the expected behaviour of replicable processes, up to various equivalences of processes/nets? **Researchers in charge of this task:** Ehrhard, Faggian, Laurent, Melliès (PPS), Hirschhoff, Hirschowitz, Sangiorgi (LIP), Regnier (IML).

TASK 5: towards a new, logically grounded, process algebra. More likely to be addressed during the last two years of the project. Use the DIN modules developed for interpreting the pi-calculus and other process algebras for defining a new process algebra which will benefit from the tight connection between DIN's and LL. For this, consider versions of DiLL possessing *additive connectives*, which will be used for expressing the *sum* operation of process algebras. Establish a real C-H correspondence for this new process algebra, with a logic which will probably be based on DiLL and explore its possible applications to process specification and certification. **Researchers in charge of this task:** all.

Scientific correspondents. Our project is clearly embedded in a French and international scientific context, and we want here to mention some of our favorite scientific correspondents. This list is of course not exhaustive: Jean-Jacques Lévy, Gérard Boudol (INRIA), George Gonthier (Microsoft Research), Catuscia Palamidessi, Dale Miller (X et INRIA), Etienne Lozes (LSV), Cosimo Laneve (Bologne),

Lorenzo Tortora de Falco (Rome 3), Stefano Guerrini (Rome 1), Kohei Honda (Queen Mary & Westfield College, Londres), Nobuko Yoshida (King's College, Londres), Robin Milner, Glynn Winskel, Martin Hyland, Marcelo Fiore (Cambridge)...

Bibliography

- [AbadiGordon] Abadi and Gordon. *A Calculus for Cryptographic Protocols : The Spi Calculus*. Proceedings of the Fourth ACM Conference on Computer and Communications Security, pages 36--47, 1997.
- [Abramsky-Jagadeesan-Malacaria] S. Abramsky, R. Jagadeesan and P. Malacaria. *Full abstraction for PCF*. Information and Computation, 163(2): 409--470, 2000.
- [Beffara1] E. Beffara. *A concurrent model for linear logic*. Proc. Mathematical Foundations of Programming Semantics MFPS XXI, 2005.
- [BergerHondaYoshida1] M. Berger, K. Honda and N. Yoshida. *Strong Normalisability in the Pi-calculus*, updated version to appear in Information and Computation, 2003.
- [BerryCurien1] G. Berry and P.-L. Curien. *Sequential algorithmes on concrete data structures*. Theoretical Computer Science 20: 256-321, 1982.
- [Blass1] A. Blass. *A game semantics for linear logic*. Annals of Pure and Applied Logic 56: 183-220, 1992.
- [Boudes1] P. Boudes. *Projecting games on hypercoherences*. Proceedings of ICALP '04, volume 3142 of LNCS. Springer, 2004.
- [Cardelli1] Luca Cardelli, web page <http://www.luca.demon.co.uk/>
- [CurienFaggian1] P.-L. Curien and C. Faggian. *L-Nets, Strategies and Proof-Nets*. Proc. Computer Science Logic 2005: 167-183
- [DanosJoinetSchellinx] V. Danos, J.-B. Joinet and H. Schellinx. *A new deconstructive logic: linear logic*. Journal of Symbolic Logic 62(3): 755-807, 1997.
- [DanosRegnier1] V. Danos and L. Regnier. *The structure of Multiplicatives*. Archive of Mathematical Logic 28: 181-203, 1989.
- [Ehrhard] T. Ehrhard. *Finiteness spaces*. Mathematical Structures in Computer Science 15 (4): 615-646, 2005.
- [EhrhardRegnier1] T. Ehrhard and L. Regnier. *The differential lambda-calculus*. Theoretical Computer Science 309: 1-41, 2003.
- [EhrhardRegnier2] T. Ehrhard and L. Regnier. *Differential interaction nets*. Theoretical Computer Science, à paraître.
- [EhrhardLaurent1] T. Ehrhard and O. Laurent. *Embedding the Finitary Pi-calculus in Differential Interaction Nets*. 2006. Electronically published proceedings of the Higher Order Rewriting workshop, HOR2006.
- [EhrhardLaurent2] T. Ehrhard and O. Laurent. *Interpreting a finitary pi-calculus in differential interaction nets*. Technical report, PPS. <http://www.pps.jussieu.fr/~ehrhards/pub/processes2.ps>
- [Girard1] J.-Y. Girard. *Linear Logic*. Theoretical Computer Science 50: 1-102, 1987.
- [Girard2] J.-Y. Girard. *A new constructive logic: classical logic*, Mathematical Structures in Computer Science 1(3): 225-296, 1991.
- [Girard3] J.-Y. Girard. *Geometry of interaction II: Deadlock Free Algorithm*. Lecture Notes in Computer Science 417: 76-93, 1988.
- [Girard4] J.-Y. Girard. *Geometry of interaction IV : the Feedback Equation*. Preprint.
- [Girard6] J.-Y. Girard. *Locus Solum*. Mathematical Structures in Computer Science 11(3): 301—506, 2001.
- [HylandOng] M. Hyland and L. Ong. *On Full Abstraction for PCF*, Information and Computation 163(2): 285-408, 2000.
- [JensenMilner] O. Jensen and R. Milner. *Bigraphs and mobile processes* (revised), Technical Report TR580, University of Cambridge Computer Laboratory, 2004.
- [JoyalNielsenWinskel1] A. Joyal, M. Nielsen, and G. Winskel. *Bisimulation from Open Maps*. In LICS '93 special issue of Information and Computation, 127(2):164-185, June 1996.
- [Krivine] J.-L. Krivine. *Dependent choices, 'quote' and the clock*. Theoretical Computer Science 308,: 259-276 (2003).
- [Lafont] Y. Lafont. *From proof nets to interaction nets*. Advances in Linear Logic, Cambridge University

Press, p. 225-247, 1995.

[Laurent] Olivier Laurent. *Polarized proof-nets and the lambda-mu calculus*. Theoretical Computer Science 290(1):161-188. (2003).

[Mazza] D. Mazza. *Multipoint Interaction Nets and Concurrency*, Proceedings of CONCUR 2005, LNCS 3653, 2005.

[Mellies1] P.-A. Melliès. *Asynchronous games 2: The True Concurrency of Innocence*. Theoretical Computer Science 358(2006), 200-228.

[Mellies2] P.-A. Melliès. *Asynchronous games 3: An Innocent Model of Linear Logic*. In proc. Category Theory in Computer Science, CTCS 2004, Copenhagen.

[MilnerSangiorgi] Milner and Sangiorgi. *Barbed Bisimulation*. In SLNCS 623: Proceedings of ICALP 92. Springer-Verlag, 1992.

[RathkeSassoneSobocinski] Rathke, Sassone and Sobocinski. *Semantic Barbs and Biorthogonality*. October 2006, submitted.

[SangiorgiWalker] Davide Sangiorgi and David Walker, *The pi-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[Winskel] M. Nygaard and G. Winskel, Domain Theory for Concurrency, Theoretical Computer Science 316: 153-190, 2004.