

	Programme BLANC	Réservé à l'organisme gestionnaire du programme N° de dossier : ANR-08-XXXX-00 Date de révision :
	Document scientifique associé	Edition 2008

Acronyme/short title

ComplICE

Titre du projet
(en français)

Complexité Implicite, Concurrence et Extraction

Titre du projet/Proposal title

Implicit Computational Complexity, Concurrency and Extraction.

(en anglais)

Les pages seront numérotées et l'acronyme du projet devra figurer sur toutes les pages du document en pied de page. Un sommaire du document est bienvenu

S'il s'agit d'un projet déposé dans le cadre d'un accord de coopération internationale*, préciser avec quelle agence étrangère :

National Natural Science Foundation of China (NSFC)
Japan Society for the Promotion of Science (JSPS)
Japanese Science and Technology Agency (JST)
National Science Council of Taiwan (NSC)

**** Veuillez vous reporter aux modalités de soumission particulières pour chaque agence sur le site de l'ANR.***

1. Programme scientifique / Description du projet ***Technical and scientific description of the proposal***

1.1 Problème posé/Rationale (1/2 page maximum)

Présentation générale du problème qu'il est proposé de traiter dans le projet et du cadre de travail dans lequel il sera effectué.

Formal methods (type systems, proof assistants ...) allow to certify various properties of software: absence of runtime errors, correctness with respect to a specification, termination ... However a crucial aspect of program behaviour is their usage of resources: execution time, memory space ... Expressing and verifying resource-related properties using proof-assistants or dedicated type systems can be difficult. Moreover one often needs to establish bounds on resource-usage that are parametric in the size of the input. This corresponds to the *complexity* properties of the programs. For instance we might want to check that a program runs in polynomial time w.r.t. the size of its input, or in logarithmic space. Using the standard on-the-shelf types and proof systems, reasoning methods... for that often requires encodings and ad-hoc techniques, which is problematic. Therefore it seems that revisiting the semantical and logical foundations of these methods should allow for the design of better adapted techniques. The study of programming languages and calculi for complexity-bounded computation is precisely the topic of *implicit computational complexity (ICC)*. The problem addressed here is the use and development of ICC to various areas: types and static methods for programming languages; program extraction from proofs; concurrent systems.

1.2 Contexte et enjeux du projet/Background, objective, issues and hypothesis (1 à 3 pages maximum)

Décrire le contexte en dressant un état de l'art national et international incluant les références nécessaires et préciser les enjeux scientifiques du projet.

- **Overview of implicit computational complexity :**

Implicit computational complexity (ICC) studies languages and calculi for resource-bounded computation, corresponding to complexity classes like for instance Ptime (programs running in time polynomial w.r.t. the size of the input), Pspace, Logspace etc. This is done either by defining

restricted languages (e.g. with conditions on control structures, operations on data) for which all programs have a given complexity bound, or by considering a general language and specifying a criterion on programs which implies that they satisfy the complexity property.

ICC has been developed since the early 90s, and has grown from two initial research lines:

- **Ramified recursion:** This line, with the works of Bellantoni-Cook and Leivant, defined restrictions of the primitive recursion scheme based on a notion of ramification of data (meant to forbid certain nestings of recursion) which allowed to characterize complexity classes like Ptime or Pspace.
- **Linear logic (LL):** this direction fits in the proofs-as-programs (Curry-Howard) paradigm and uses linear logic which is a system giving a logical status to duplication thanks to specific connectives. Following initial ideas of Girard, variants of LL, called light logics, can be defined, in which execution (normalization of proofs) corresponds to Ptime computation:

BLL [GSS92], LLL [Gir98], SLL [Laf04]. Some of these systems have been later adapted into type systems for lambda-calculus, the core of functional programming languages, with the following property: if a program is well-typed, then it is Ptime.

A practical limitation of these two approaches is their modest *intensional expressivity*. Even if for instance all Ptime functions can be represented by a given ICC system, often some common algorithms are not validated: for instance the sorting function on lists can be represented, but usually not with the Merge sort or Quicksort algorithms. This observation has triggered the design of more flexible ICC systems:

- on the one hand the line of work on ramified recursion has led to the approach of *quasi-interpretations* [MM00,BMM05] for first-order functional programs.
- On the other hand linear type systems for non-size-increasing computation [Hofmann03] have been proposed for lambda-calculus based languages.

Beside the functional paradigm, more recently some ICC criteria have also been designed for some core imperative languages, in particular by interpretation methods based on the study of the size of values [JK05,NW06].

At the international level, the vitality of research in ICC is in particular illustrated by the fact that papers on this topic are regularly presented at the main international conferences of the domains involved (IEEE LICS, CSL, TLCA ... for instance at least 6 papers at LICS since 2004).

- **International actors of the domain:**

- **Munich (Germany):** this site is a leading one in this area with in particular the contributions of M.Hofmann, H. Schwichtenberg, U. Schöpp. It contributes both to the foundational aspects (LL and realizability methods; type systems; characterization of complexity classes, in particular small classes with the project ProPlatz) and to more applied ones, like through the european projects MRG (Mobile Resource Guarantees) and Embounded, for which Munich has played a key role.
- **Turin, Bologna (Italy):** these two italian sites have strong competences in types and lambda-calculus and have contributed to ICC through the approach of LL, type systems and inference, realizability methods (Dal Lago, Martini, Ronchi Della Rocca, Gaboardi).

In Rome there are also contributions to the LL approach through the angle of proof-nets and semantics (Tortora de Falco, Pagani). These works have been supported in particular by successive Italian national projects: Protocollo, Follia.

- **Nancy:** the group of Marion at LORIA is an influential site with the works on ramification and quasi-interpretation methods for first-order functional programs, partly inspired by the tradition of term-rewriting systems (polynomial interpretations for termination), and ICC for other models of computation (Blum-Shub-Smale BSS model) (Marion, Bonfante, Bournez, Péchoux). More recent developments include the work on sup-interpretations, refining that of quasi-interpretations.
- **Paris:** the site of Paris 13 has contributed in this area to LL and typing approaches, quasi-interpretations, BSS complexity, and that of Paris 7 to LL, semantics, quasi-interpretations and their application to synchronous systems. The group of Paris 13 has organized three international workshops on ICC in 2004, 2006 (with J.-Y. Marion) and 2008, which have gathered the international community working on this topic.

Besides these groups there are also influential individual researchers: Terui (Tokyo) on linear logic and typing approaches; Kristiansen (Oslo) and Niggel (Ilmenau, Germany) on matrix-based ICC methods for imperative programs, Mairson (Brandeis, US) on linear logic approaches, Leivant (Bloomington, US) on ramification and logical approaches.

- **The NOCoST project (ANR young investigators project):**

This project between Paris 13 and Paris 7, continuing until 12/2008, has as goals to investigate the semantic foundation of ICC and the development of ICC systems based in particular on linear logic and type systems, but also on interpretation methods. It has already contributed to establishing Paris as a visible site in the ICC community, in particular with publications in leading conferences (IEEE LICS, CSL), collaborations with foreign researchers (Terui, Dal Lago, Coppola) and organization of workshops. As examples of notable contributions let us mention the semantic techniques for ICC (coherence spaces [LTdF06], geometry of interaction [BCDL07]), the type inference methods for LL-based type systems [ABT06,ABT07], the boolean circuits characterization of complexity classes by means of linear logic [MR06]. The project has been fostered by the collaborations with Postdoctoral young researchers (Dal Lago, Mazza, Moyen) and several visits of international researchers of this area, as well as of foreign PhD students (8 visits since 2006).

Web page of the project:

http://www-lipn.univ-paris13.fr/nocost/rubrique.php3?id_rubrique=5

The most recent international workshop on ICC has been organized by the participants of NOCoST (organizers: P. Boudes, V. Mogbil, P. Jacobé de Naurois) at Univ. Paris 13 in February 2008, with around 40 participants (from 10 countries) and 23 talks, making it an important meeting of this community.

<http://www-lipn.univ-paris13.fr/~mogbil/wicc08/>

- **Other european context information:**

A proposal of european project called CARA (Coordinated activities in resource analysis) led by Nijmegen Radboud University (M. Van Eekelen) has been submitted in january 2008 to the FET FP7 call. It includes 17 sites (in particular Paris 13, Paris 7, Nancy, Munich, Bologna...) and plans to foster research on various aspects of resource analysis for programs, with in particular a foundational side partly related to ICC. If accepted this project will organize regular workshops and fund travels of researchers between sites for collaborations.

- **Current scientific issues:**

Despite the progresses obtained in ICC during the last years, note that this approach has been mainly developed and analysed in the functional paradigm, and that there remain several open issues:

- the criteria obtained are still not flexible enough (intensional expressivity) and efficiently checkable in order to be usable for real-size common programs;
- the issue of applying these techniques to programs obtained by extraction from formal proofs (such as those extracted from Coq proofs) has not been really explored yet;
- the area of process calculi is important for modelling concurrent systems and is an active research field, but ICC methods are not yet available for this area, apart from the specific setting of synchronous systems (synchronous pi-calculus of Amadio).

Note also that even if there have been some interesting recent works on small complexity classes (Logspace [Sch07], NC [BKMO06, Ter04]) these are not characterized in a way as satisfactory as the class Ptime, since the calculi used are up to now significantly more complicated.

1.3 Objectifs et caractère ambitieux/novateur du projet/Specific aims, highlight of the originality and novelty of the project (1 à 2 pages maximum)

Décrire les objectifs scientifiques/technologiques du projet.

Présenter l'avancée scientifique attendue. Préciser l'originalité et les ambitions du projet.

Détailler les verrous scientifiques et technologiques à lever par la réalisation du projet.

This project aims at developing implicit computational complexity and its usage for certifying complexity bounds, along the following main directions:

- functional programming: by improving current ICC systems and combining together different ICC criteria we plan to develop systems for functional programs which are more flexible;
- extraction from formal proofs of programs with bounded complexity;
- concurrent systems: adapting the methods of ICC to the setting of synchronous systems (a direction which is already initiated) and to that of process calculi like the Pi-calculus, in order to guarantee statically some quantitative properties.

The two last points have stayed up to now largely unexplored, but they would open the perspective of spreading ideas and results of ICC to two areas and communities for which the quantitative aspects of resource-usage are particularly relevant and motivating.

We aim in particular at developing an experimental core proof-assistant for a second-order logic coming from ICC, with extraction of certified Ptime programs of lambda-calculus.

We will also design type systems or static criteria ensuring various properties on Pi-calculus processes, such as possibly: termination; termination with a polynomial number of steps (w.r.t. the size of the initial system); execution with a polynomial bound on the size of the system.

For these goals we will take advantage of the progress made on semantical approaches to ICC (with e.g. tools from geometry of interaction, game semantics) and logical methods (proof-nets), as developed in particular in the previous NOCoST ANR project (running until 12/08).

These tools should in particular enable us to find solutions to the main obstacles towards the goals we mentioned. On the side of proof extraction the main difficulty is that of proof-search in ICC-oriented logical systems, which up to now was problematic. On the side of concurrent systems one of the main issues is to lift techniques initially designed for sequential functional programs to interactive processes (with dynamic I/O flow): for that the point of view of semantics and linear logic should provide important insights.

With this project we also aim at gathering the main groups currently working on ICC in France, in order to consolidate our international position, and to be able to take part to European-level initiatives.

1.4 Description des travaux : programme scientifique/For each specific aim: a proposed work plan should be described (including preliminary data, work packages and deliverables) (10 pages maximum)

Décrire le programme de travail décomposé en tâches en cohérence avec les objectifs poursuivis Les tâches représentent les grandes phases du projet. Elles sont en nombre restreint.

Pour chaque tâche, préciser :

- *les objectifs de la tâche*
- *le programme détaillé des travaux correspondants.*

Background and context.

* Implicit computational complexity criteria and methodology..

Implicit computational complexity (ICC) studies either languages for which all programs have a given complexity bound, or, in the setting of a general programming language, some criteria implying that a program has a given complexity, for instance Ptime. Often this is essentially a matter of presentation (both viewpoints can be adopted). Along the years, to the initial goal of providing machine-free characterizations of complexity classes such as Ptime or Pspace, ICC has also added the goal of designing criteria allowing to statically certify that a program admits a certain complexity bound. Note that the property of being Ptime, or of another complexity, is undecidable, so with a decidable criterion one can only expect a sufficient condition.

The progresses of ICC have been along several distinct axes: from the characterization of Ptime to that of other classes (Pspace, Logspace, NC...); from the setting of functional programming to other paradigms (imperative programs, synchronous languages...); from constrained calculi to languages which are intensionnally more expressive and allow to represent interesting algorithms. In this project we aim at pursuing objectives in these three directions.

It can be observed that often ICC criteria can be decomposed into two parts: on the one hand a condition ensuring termination (recursive path ordering, 'stratification' of light logics) and on the other hand a condition providing a bound on the size of values (quasi-interpretation; condition on duplicable terms for light logics). This decomposition has then been used as a methodology to design new ICC criteria, as for instance in the recent works on imperative LOOP-languages like [NW06,JK05], in which the authors define some certificates ensuring bounds on values, that can then be exploited to obtain complexity bounds.

* The linear logic approach.

Linear logic is a system refining classical logic by giving a logical status to duplication of hypothesis [Gir87], and which is thus resource-conscious. It fits in the proofs-as-programs correspondence: formulas correspond to types and normalization of proofs (or cut elimination) to program execution.

An important tool of linear logic is that of *proof-nets*, a representation of proofs by graphs, which allows to see normalization as a graph-rewriting process. Proof-nets have opened the way to a geometrical point of view on computation,, in particular to *Geometry of interaction* [Gir88], a programme initiated by Girard aiming at interpreting proof normalization as a flow of information in the proof-net. For instance, one possible model of geometry of interaction consists in seeing the proof-net as an automaton acting on tokens, and execution is performed by computing the paths followed by the tokens.

Coming back to normalization, as the main source of complexity in normalization is duplication, and as duplication is regulated in linear logic by specific modalities (denoted $!$, $?$), this triggered the idea of defining variants of the modalities for which normalization/execution would have a tamed complexity. This line of research has led to the introduction of the systems BLL [GSS92], LLL [Gir98], SLL [Laf04], dubbed *light logics* and which characterize Ptime computation. In these systems normalization can be performed in Ptime, and conversely any Ptime function can be represented.

Light logics have then been used as type systems for lambda-calculus, the core of functional programming languages [Bai04,BT04,GR07]. The idea is that if a lambda-calculus program is well-typed, then it is of Ptime complexity. Deciding if a program admits a type in such a system is therefore a way to statically obtain a complexity bound. Type inference algorithms have been developed for this purpose [ABT07,BT05,CR05].

* Concurrent system and process calculi.

Areas like mobile computing, communications protocols, real-time systems cannot be modelised using sequential calculi and require specific formalisms. Like lambda-calculus for sequential computing, process calculi are low level target languages for the foundations of concurrent systems and mobile computing.

Processes are an abstraction of threads of control. Different desirable observable behaviors, such as communication discipline, interruptions, or processes migration, lead to different formalisms. The most well known representatives of process calculi are CCS (Calculus of communicating systems, [Mil89]) and the pi-calculus [MPW92], where processes communicate by messages on named channels (Each communication involves a rendez-vous between an emitter and a receiver process).

A related area is that of concurrent systems focusing on reactivity (synchronous programming); it can be modelised by the synchronous pi-calculus [Ama07]. A program in the synchronous pi-calculus is composed by a collection of parallel threads, interacting through signals, and whose computation is regulated by instants (or rounds, or phases). At the beginning of each instant, a collection of signals is available in input and at the end of each instant a collection of signals is available in output. A thread can also react (at the next instant) to the absence of a signal within the current instant.

Following the time bounds obtained in [AD07] for the synchronous pi-calculus, we think that synchronous languages are an interesting candidate for extensions of ICC techniques to a richer programming framework than that of plain sequential functional programming. Indeed on the practical side, synchronous languages are often used to program embedded systems, an application area where resource control plays an important role. On the theoretical side, one has to cope with the fundamental concepts of parallelism, concurrency, and interaction.

Research Programme.

Our research programme is organized in 3 Parts, corresponding to 5 tasks. We describe these tasks and their objectives below.

Part I. Semantic and logic foundations.

ComplICE

*** Task 1: Develop semantic and logic-based account of implicit complexity**

(LIPN: Baillot, Boudes, Mogbil / LIP: Laurent / LORIA: Bonfante, Guiraud, Marion)

Developments based on linear logic are important for implicit characterizations of complexity classes [Gir98,Laf04]. Current directions concern unification of the various light systems (in particular by means of more abstract and semantical tools), logical characterizations of additional complexity classes, and extension of the logical ICC approach to other geometric models of computation.

- 1.a Denotational and game semantics:

As several variations on the theme of light logics are available, it seems that the foundations behind these systems are not properly understood yet. We aim to pursue the analysis of these logics, using in particular semantic tools, in order to simplify them and to obtain a more general framework subsuming the currently still incomparable systems. Games and denotational semantics will be useful here by providing a semantic setting in which the various calculi and the quantitative aspects of their interactions can be studied, while taking some distance with the syntactic details of the logics. This needs to go behind the results obtained from static models [Bai04,LTdF06,dCPTdF08].

We aim in particular at defining a game category whose morphisms would be strategies satisfying certain conditions, defined semantically, such that the strategies acting on data-types would all correspond to Ptime algorithms. Actually a game semantics for LLL has already been given by Murawski and Ong [MO00], but our goal differs from their work in the sense that we would like the complexity property to be expressed directly on the strategies. Moreover a game semantics of this kind could provide a sound model of several light logics (not only LLL), and we would also expect it to contain more strategies than just those defined by the logics. In this case one could then use the semantics as a solid basis to define extensions of the logics keeping the Ptime property, but more expressive intensionally. In particular this would be a good setting to analyse how programming features such as exceptions or local references could be added to light logics while preserving the complexity bounds.

- 1.b Linear logic by levels:

We plan to continue the work on light linear logic by levels (L4) [BM07], a system designed to simplify the proof-nets of LLL. In particular L4 does not use the §-boxes of LLL, but still has, like LLL, § modalities occurring in formulas. In the same paper [BM07], a variant L4_0 without § modalities (but only a notion of level on atomic formulas) was also introduced. However L4_0 does not offer all the properties expected. We would thus like to design a light logic which would have similar complexity properties as LLL or L4, still would not use any explicit extra modality but instead just a notion of level on atomic formulas. For this goal it might be useful to consider the point of view of geometry of interaction [Gir88,DR95] which could help in simplifying the logic and the corresponding proof-nets. This line of work can be developed in collaboration with D. Mazza (Postdoc at Paris 7 in 2008).

• 1.c A light logic for complexity in space:

The goal here is to define a logical system characterizing programs efficient in terms of memory space needed for their execution. More precisely this efficiency corresponds to algorithms handling a constant number of pointers (constant w.r.t. the size of the input): they implement the functions Complice

computable in logarithmic space, which is the class Logspace. A variant of linear logic for Logspace has been given by [Sch07], based on some ideas of BLL [GSS92], but it has the drawback of handling a complicated syntax with in particular some explicit annotations of polynomials in the types. In this system polymorphism is bounded and higher-order functions are used for representing efficient algorithms. Besides, another characterization of a space complexity class by a variant of linear logic has been obtained in [GMR08] with a language for Pspace, based on the system SLL [Laf04].

We aim at defining a logic characterizing Logspace based on the approach of SLL and which hopefully could be simpler than the previous one of [Sch07]. Indeed as various type systems defined as variants of SLL have allowed to characterize different time and space complexity classes (such as Ptime, Pspace, Nptime) it seems possible that this approach could also be tuned for the smaller class Logspace. For that we might also use ideas from other ICC characterizations of Logspace obtained by restrictions of primitive recursion [Nee04,Kri05].

- 1.d Polygraphic approaches to ICC

The computational model underlying light logics is mainly given by proof-nets. Related geometric approaches to computation (such as Lafont's interaction nets or Burroni's polygraphs) can also be used to give implicit characterizations of complexity classes. Bonfante and Guiraud showed in [BG07a] that polygraphs have good intensional properties. In particular, divide-and-conquer algorithms fit in a very natural way the theory of current/heat interpretation as introduced by Guiraud in his thesis. From this work, Bonfante and Guiraud revealed the intrinsic functorial and the derivational properties of current and heat interpretation [BG07b]. Among open problems, we mention the question of strategy of computation. Since polygraphs offer a very nice algebraic framework to deal with them, we expect to get some new characterization of complexity classes with even more intensionality.

Objectives:

- (1.1) Resource-aware game semantics (18 months)
- (1.2) Semantical characterizations of complexity classes (24 months)
- (1.3) Levelled light logic for Ptime (12 months)
- (1.4) Light logics for space complexity classes (18 months)
- (1.5) Polygraphic characterizations of complexity classes (18 months)

Possible international collaborators:

U. Dal Lago (Bologna), M. Gaboardi (Torino), U. Schopp (Munich)

Part II. Applications to the functional setting: typing, program extraction

Task 2: Types and criteria for bounding the complexity of programs

(LIPN:Baillot, Jacobé de Naurois, Moyen/ LORIA: Bonfante, Marion, Pechoux)

2.a Interpretation methods:

The issue covered by this approach is to develop criteria of complexity bounds for some programming languages without testing nor monitoring.

As a paradigmatic example, the method of quasi-interpretations has shown its strengths in the last years in the domain. Recall that it is inspired by the technique of polynomial interpretations, a classical tool in the study of termination, but with more lax conditions. A quasi-interpretation for a program guarantees bounds on the sizes of values occurring during computation. In the setting of first-order functional programs, quasi-interpretations combined with recursive path orderings provide complexity criteria characterizing various complexity classes such as Ptime, and Pspace [BMM07,BMM05,MM00].

Among the conditions of success one is hoping for in such an approach,, we can mention:

- the fact that the complexity criterion has good intensional properties. That is, that it validates a large --actually as large as possible-- class of programs.
- the fact that the criterion can be effectively tested. As an illustration, using the procedure due to Tarski, one shows that, given a program, it is possible to decide the existence for this program of a quasi-interpretation over the reals (assuming a bounded degree).

Until now, we essentially considered first-order functional programs. Methods by interpretation can undoubtedly be enlarged to wider programming frameworks. Generalisation to higher order programs is a natural goal, but maybe more promising are generalisations to imperative programming or object oriented programming style. On this latter subject, work by P echoux and Marion [MP07] have already given some promising first results and should be developed.

On another level, methods by interpretation as mentioned here were originally termination proof methods, but it is clear that there should be some connections with the abstract interpretation framework developed by Cousot. Better views on the subject would certainly broaden the understanding of our methods.

A software, called CROCUS, has been developed at the LORIA. It computes quasi-interpretations of programs. But presently, it does not combine this method with other approaches of implicit complexity theory such as path orderings, matrix abstract evaluation, etc. Another valuable point would be to make CROCUS output explicit complexity bounds rather than quasi-interpretations.

2.b Type systems

This subtask will possibly build on results from Task 1 and techniques from 2.a.

Following previous works on the design of type systems for lambda-calculus based on light logics [BT04,GR07] we want to define a new one coming from light linear logic by levels L4 [BM07] or from a variant of this system. Indeed a good property for such a system is to have a simple language of types, and for that L4_0 (without § modality) or a similar system arising from the task 1.2 would be a good candidate. It would then allow for a simpler and perhaps more efficient type inference algorithm than the one of [ABT07] for DLAL. This work can be pursued in collaboration with D. Mazza (Paris 7).

Moreover, as the type systems derived from light logics are currently available only for lambda-calculus and only have a limited intensional expressivity, we would also like to investigate them through a more pragmatic approach. For that we will try to combine them with the quasi-interpretations approach in a single analysis. The idea is to consider a core functional language with a type system based on light logics and possibly some extra annotations corresponding to quasi-interpretations informations. In this way we would combine the advantages of quasi-interpretations (handling of recursion and pattern-matching) and that of light logics (covering higher-order and polymorphism) in a single system. The benefit would be to validate more interesting programs,

written in a sufficiently rich language. This research line might be conducted in connection with the one mentioned previously of quasi-interpretations for higher-order programs, since they tackle closely-related issues.

Objectives:

- (2.1) Extension of the quasi-interpretations method to higher-order programs (24 months)
- (2.2) Development of the quasi-interpretations analysis of object-oriented programs (18 months)
- (2.3) Enhancement of the features of the CROCUS software (36 months)
- (2.4) Design of a type system based on L4 and the corresponding type inference algorithm (12 months)
- (2.5) Combining light logics types and quasi-interpretations in a single ICC criterion. (24months)

Possible international collaborators: U. Dal Lago (Bologna), K. Terui (Tokyo).

***Task 3:** Extraction from proofs of complexity bounded programs:
(LIPN: Baillot, Fouqueré, Mayero, Moyen / LIP: Laurent)

Recall that some proof assistants like Coq or Nuprl are equipped with an extraction procedure allowing to produce a functional program realizing the algorithmic contents. Of the proof. However one has no guarantee on the complexity of the extracted program (see e.g. [Ben04]), which can for instance be of exponential time even if the original proof is simple. We want to explore the possible refinement and enhancement of proof assistants so as to allow for the extraction of feasible certified programs. Our idea is to use the principle of **programming with proofs** [KP90]: one starts with a specification, of which we prove the totality by using a particular proof system, which here would be based on a light logic, and finally we extract from the proof a lambda-calculus program. In this case the light logic would guarantee that the extracted program is of Ptime complexity. It would thus be an extraction of feasible program.

Light logics are a natural starting point in this direction, as they were precisely initially introduced by Girard together with a naïve set theory [Gir98] allowing for the extraction of Ptime programs, which has later been studied by Terui [Ter04b]. The difficulties here are reminiscent of those arising in the functional programming setting: one needs on the one hand a proof system which is expressive enough to be able to carry out proof construction in a manageable way, and on the other hand one would like the search for the modality parts of the proofs (responsible for the complexity analysis) to be automatized. The works on the simplification and generalization of light logics, and on the implementation of type inference will be important tools here. Indeed the system L4 of [BM07] gives a system with fewer sequentiality conditions than LLL (because there are no more § boxes) and which should facilitate the proof-search. The search for the modality parts of the proof could presumably be handled by techniques coming from type inference [ABT07].

We might have first to compare several proof systems in order to determine which ones are more suitable for this study. The most natural one will probably be a second-order proof system in the style of AF2 [KP90]. A first step will then be to study proof-search in this logic (starting with the propositional fragment). We will also analyse the extraction procedure and the programming-with-proofs methodology in this setting. We will then implement an interactive proof-search engine with an extraction functionality (to lambda-calculus). Another motivating, but more speculative direction, will be to investigate whether the techniques of light logics could be adapted to the

calculus of constructions (underlying Coq) [CH88]. An encouraging point for that is that light logics are quite flexible with respect to several extensions (type fix-points, naïve set theory comprehension etc.). Ideally one could hope to delineate certain classes of proofs in Coq (with some extra informations, like maybe some kind of levels) for which the program extraction would produce certified feasible programs.

Objectives:

- (3.1) analyse proof-search in L4 or another light logic (18 months)
- (3.2) define a light type system in the style of AF2 and study the properties of extraction (24 months)
- (3.4) implement a prototype proof-assistant for this light AF2 with tactics for proof-search (48 months) and extraction of lambda-term.

Possible international collaborators:

K. Terui (Tokyo), U. Dal Lago (Bologna)

Part III. New directions: concurrent systems, parallel complexity classes

- **Task 4:** ICC for process calculi

(LIPN: Amadio, Baillot, Boudes / LIP: Demangeon, Hirschhoff, Laurent)

The setting of concurrent systems extends that of sequential computation by allowing to represent networks of communicating processes evolving dynamically and non-deterministically, as in mobile computing or security protocols for instance. There is a natural need in this area for techniques to statically predict or control bounds on the use of resources, estimated for instance by the running time, the number of processes or the number of messages exchanged.

Following the insights from ICC and light logics, we aim at investigating techniques to guarantee quantitative bounds on concurrent systems, by designing suitable type systems or static criteria. For instance we would like to design criteria that would guarantee that a system will terminate after a certain number of steps (expressed in function of the size of the initial configuration), or that during execution the size of the system will remain bounded. Such properties are clearly relevant in a concurrent setting: for instance, one would like a request to a server to be answered after a reasonable number of computation steps, or that an agent does not use too many resources for too much time.

- 4.a Resource analysis for synchronous languages:

Some work in this direction has been carried out recently in the particular setting of synchronous systems by Amadio and Dabrowski [AD07], allowing to statically bound for instance the response time of processes. This has been achieved by exploiting ideas from implicit computational complexity, more precisely from the Quasi-interpretations approach for rewriting systems [MM00].

Following this work, resource analysis can address increasingly ambitious goals. A first step is to guarantee that each instant terminates. A second step is to bound the size of the computed values as a

function of the size of the parameters at the beginning of the instant. A third step is to guarantee an effective reactivity for arbitrary many instants. In particular, we have proposed a notion of feasible reactivity which informally means that there is a polynomial that uniformly for any instant bounds the reaction time of the program as a function of the largest input received at all previous instants. We have been developing static analyses that guarantee feasible reactivity. A characteristic of the analyses is that to a great extent they make abstraction of the memory and the scheduler. This means that each thread can be analysed separately, that the complexity of the analyses grows linearly in the number of threads, and that an incremental analysis of a dynamically changing system of threads is possible. Ongoing and planned work aims at simplifying the analysis and extending its applicability to a larger fragment of the language.

- 4.b Termination and Implicit complexity criteria for non-synchronous process calculi:

Recall that in the setting of first-order functional programming, ICC criteria can often be decomposed into on the one hand a condition for termination (recursive path ordering, type system) and on the other hand a condition bounding the size of values (e.g. Quasi-interpretation). Therefore it is likely that for process calculi like the pi-calculus, criteria for quantitative bounds will also need to rely on conditions for termination. We will thus develop the analysis of termination for non-synchronous models of concurrency such as the pi-calculus. Existing works on this problem exist, and can be classified into two families of approaches.

The first approach draws on the technique of logical relations, that has been developed in the sequential setting, to isolate classes of terminating processes [San06,YBH04]. We believe it would be interesting to increase the expressiveness obtained in these works, since the analysed processes correspond to essentially sequential behaviours. We would like to exploit for that the connection with semantical methods known for the lambda-calculus.

The second approach focuses on applying techniques from rewrite systems in order to define type systems that are able to capture typical programming idioms of the pi-calculus, and guarantee termination of certain processes [DS06,DHKS07]. Because of the tools it uses, this direction seems more related to the technique of quasi-interpretations. We plan to extend further the existing methods, and to study their applicability on examples (some of these type systems are already implemented in the TyPiCal tool [Kob07]).

We think that some ideas of light logics might be adaptable to the domain of concurrent calculi because the constraints of these systems enforcing complexity bounds are not tied to specific control structures or data-types from functional programming. It therefore seems plausible that the high-level structure of light logics (structural or geometrical conditions, I/O properties) could be transferred from the lambda-calculus to concurrent calculi in the style of CCS or of the pi-calculus. Possible bridges could follow interaction nets approaches to concurrency (multi-port interaction nets [Maz05], differential interaction nets [EL07]) or higher-order process algebra related with linear logic such as HOPLA [WZN04].

Another interesting challenge is to be able to express and guarantee resource control policies that address some form of ‘space usage’ of concurrent processes (limit the number of users accessing a server, or the number of subtasks spawned by a given agent, enforce some scheduling policy, etc.). Some approaches have been proposed in the framework of Mobile Ambients [TZH02, BBDCS07]. Our goal is to use the technology of light logics to provide more general solutions where, in particular, asymptotic bounds are given instead of constant bounds.

Objectives:

- (4.1) Termination criteria for process calculi (24 months)
- (4.2) Complexity bounds in the synchronous model (24 months)
- (4.3) Quasi-interpretation for the non-synchronous model (24 months)
- (4.4) Light typing systems for process calculi (36 months)
- (4.5) Space usage control for process calculi (48 months)

*Task5: Parallel languages for NC

(LIPN: Jacobé de Naurois, Mogbil / LORIA: Bonfante, Marion, Péchoux)

Implicit characterizations of complexity classes are essentially done in a sequential setting. However the smallest complexity classes --corresponding to the more efficient algorithms-- treat (logarithmic) space or parallel computation. Implicit current approaches on parallel computation are, on the one hand, a kind of Curry-Howard isomorphism in the parallel setting, and on the other hand, characterizations by means of recursion schemes. Both approaches are related and studied here:

- 5.a A language for NC starting from Boolean proof-nets:

Can we define via the Curry-Howard isomorphism some implicit complexity calculi for the class of efficiently parallelizable algorithms (class NC, defined thanks to boolean circuits) ? This question can be addressed by defining calculi associated to boolean proof-nets from linear logic [Ter04,MR06]. In fact, such uniform boolean proof-nets characterize NC by using a parallel cut-elimination procedure.

We aim at investigating techniques to parallelize reduction steps for a suitable language, by managing critical pairs. For instance the elimination of a sequence of cut axioms could correspond in this way to a redex of arbitrary size but of a very specific shape which accounts for the acceleration due to the parallelism of the computational model. We are thinking of calculi where reduction steps would be triggered in parallel between redexes of same shape, as is already the case in boolean proof-nets.

The characterized behaviours in concurrent computation lead to a large class of formal systems called process calculi. We plan to study relationships between the design language and models of process calculi. For example we shall pay attention to aspects corresponding to the behaviour of the boolean proof-nets which use essentially higher-order functions.

- 5.b Theoretical recursion schema for the parallel classes:

In another framework, that of the recursion schemes approach, the question 5.a has already been already answered positively by use of restricted recursion schema, see for instance the work of P. Clote. However, he uses an explicit size bound, à la Cobham, to ensure the correctness of his schema. The approach has been renewed in [BKMO06] where a characterization of NC is given.

In [BMP06], Bonfante, Marion and Péchoux have shown that result obtained for theoretical schema can be transported smoothly to more intensional criteria such as sup-interpretations (a generalization of quasi-interpretations). For this kind of work, recursion schema serve as a guideline. Among open

questions is the case of AC, the class of circuits with unbounded fan-in gates. It is well known that the hierarchies of the NC^k and AC^k are in 'sandwich' but there is no convincing equivalent construction at the recursive theoretic level. Another goal is to broaden the work in [BMP06] to get more intensionality and to enlarge the target programming languages.

As the recursion approach to ICC has been related to linear logic in the case of sequential computation (characterizations of Ptime), this approach might provide hints for our first goal.

Objectives:

- (5.1) Language for NC based on Boolean proof-nets (30 months)
- (5.2) Relationship between this language and process calculi (48 months)
- (5.3) Recursion schema for AC (18 months)
- (5.4) Recursion theory for the parallel classes (48 months)
- (5.5) Extension of current programming languages (24 months)

Possible international collaborators: K. Terui (Tokyo).

References.

- [ABT06] V. Atassi, P. Baillot, K. Terui. Verification of Ptime Reducibility for system F Terms via Dual Light Affine Logic. In proceedings of CSL'06, LNCS 4207. Springer 2006.
- [ABT07] V. Atassi, P. Baillot, K. Terui. Verification of Ptime Reducibility for system F Terms: Type Inference in Dual Light Affine Logic. Logical Methods in Computer Science, 3(4:10):1-32, 2007.
- [AD07] R. Amadio and F. Dabrowski. Feasible reactivity in a synchronous pi-calculus. In proceedings of PPDP'07, ACM 2007.
- [Ama07] Roberto Amadio, A synchronous pi-calculus, Journal of Information and Computation 205, 9 1470-1490, 2007.
- [Bai04] P. Baillot. Type inference for Light Affine Logic via constraints on words. Theoretical Computer Science, 328(3):289--323, 2004.
- [BBDCS07] F. Barbanera, M. Bugliesi, M. Dezani-Ciancaglini, V. Sassone. Space-aware ambients and processes. Theor. Comput. Sci. 373(1-2):41-69, 2007.
- [BC92] S. Bellantoni, S. A. Cook: A New Recursion-Theoretic Characterization of the Polytime Functions. Computational Complexity 2: 97-110, 1992.
- [BCDL07] P. Baillot, P. Coppola, U. Dal Lago. Light Logics and Optimal Reduction: Completeness and Complexity. In Proceedings of LICS'07. IEEE Computer Society Press, 2007.
- [Ben04] R. Benzinger: Automated higher-order complexity analysis. Theor. Comput. Sci. 318(1-2): 79-103, 2004.
- [BG07a] G. Bonfante and Y. Guiraud. Intensional properties of polygraphs. Proceeding of Termgraph'07, ENTCS to appear. 2007
- [BG07b] G. Bonfante and Y. Guiraud. Polygraphic programs and polynomial-time functions. Logical Methods in Computer Science, 2007.
- [BKMO06] G. Bonfante, R. Kahle, J.-Y. Marion, and I. Oitavem. Towards an implicit characterization of NC^k . In Proceedings of CSL'06, LNCS 4207. Springer, 2006.
- [BM07] P. Baillot and D. Mazza. Linear Logic by Levels and Bounded Time Complexity. Submitted, 2007.

- [BMM05] G. Bonfante and J.-Y. Marion and J.-Y. Moyen. Quasi-Interpretations and Small Space Bounds, Proceeding of RTA'05, LNCS 3467. Springer 2005.
- [BMM07] G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. Quasi-interpretations, a way to control resources. Theoretical Computer Science, to appear.
- [BMP06] G. Bonfante, J.-Y. Marion and R. Pechoux. A characterization of alternating log time by first order functional programs. In Proceedings of LPAR'06, LNCS 4246. Springer, 2006.
- [BT04] P. Baillot, K. Terui. Light types for polynomial time computation in lambda-calculus. In Proceedings of LICS'04. IEEE 2004.
- [BT05] P. Baillot and K. Terui. A feasible algorithm for typing in Elementary Affine Logic. In Proceedings of TLCA'05, LNCS 3461. Springer 2005.
- [CH88] Th. Coquand and G. Huet. The Calculus of Constructions. Information and Computation, 76 (2/3), 1988.
- [CR05] P. Coppola, S. Ronchi Della Rocca: Principal Typing for Lambda Calculus in Elementary Affine Logic. Fundam. Inform. 65(1-2): 87-112, 2005.
- [dCPTdF08] D. De Carvalho, M. Pagani, and L. Tortora De Falco. A semantic measure of the execution time in linear logic. Submitted to Theor. Comput. Sci.
- [DHKS07] R. Demangeon, D. Hirschhoff, N. Kobayashi, and D. Sangiorgi. On the complexity of termination inference for processes. In Proceedings of TGC'07, LNCS. Springer, 2008. to appear.
- [DL06] U. Dal Lago: Context Semantics, Linear Logic and Computational Complexity. in Proceedings of IEEE LICS 2006: 169-178, IEEE Computer Society, 2006.
- [DR95] V. Danos and L. Regnier. Proof-Nets and the Hilbert Space. Advances in Linear Logic, LMSLNS 222, Cambridge University Press 1995.
- [DS06] Y. Deng and D. Sangiorgi: Ensuring termination by typability. Inf. Comput. 204(7): 1045-1082, 2006.
- [EL07] T. Ehrhard and O. Laurent. Interpreting a Finitary Pi-Calculus in Differential Interaction Nets. In Proceedings of CONCUR'07, LNCS 4703. Springer 2007.
- [G07] M. Gaboardi. Linearity: an Analytic Tool in the Study of Complexity and Semantics of Programming Languages. PhD thesis, University of Turin/ INPL Nancy, December 2007.
- [Gir87] J.Y. Girard. Linear Logic, Theoretical Computer Science, vol. 50. 1987.
- [Gir88] J.-Y. Girard. Geometry of Interaction I: an Interpretation of System F. In Proceedings of the Logic Colloquium '88, North-Holland, 1988.
- [Gir98] J.-Y. Girard. Light Linear Logic. Information and Computation 143(2): 175-204, 1998.
- [GMR08] M. Gaboardi, J.-Y. Marion and S. Ronchi Della Rocca. A Logical Account of PSPACE. In Proceedings of POPL'08. ACM, 2008.
- [GR07] M. Gaboardi and S. Ronchi Della Rocca. A soft type assignment system for lambda-calculus. Proceedings CSL'07, volume 4646 of LNCS, pages 253--267, Springer, 2007.
- [GSS92] J.-Y. Girard, A. Scedrov and P. J. Scott. Bounded Linear Logic: A Modular Approach to Polynomial-Time Computability. Theor. Comput. Sci. 97(1):1-66, 1992.
- [Hof00] M. Hofmann: Safe recursion with higher types and BCK-algebra. Ann. Pure Appl. Logic 104(1-3): 113-166, 2000.
- [Hof03] M. Hofmann: Linear types and non-size-increasing polynomial time computation. Inf. Comput. 183(1): 57-85, 2003.
- [Jon99] N. D. Jones: LOGSPACE and PTIME Characterized by Programming Languages. Theor. Comput. Sci. 228(1-2): 151-174, 1999.
- [JK05] L. Kristiansen and N. D. Jones. The flow of data and the complexity of algorithms. Cooper, L.Åwe, Torenvliet (eds.), CiE'05:New Computational Paradigms, LNCS 3526. Springer 2005.
- [Kob07] N. Kobayashi. TyPiCal: Type-based static analyzer for the Pi-Calculus. available from <http://www.kb.ecei.tohoku.ac.jp/~koba/typical/>, 2007.

- [KP90] J.-L. Krivine, M. Parigot: Programming with Proofs. *Elektronische Informationsverarbeitung und Kybernetik* 26(3): 149-167, 1990.
- [Kri05] L. Kristiansen. Neat function algebraic characterizations of logspace and linspace. *Computational Complexity* 14(1):72-88, 2005.
- [Laf04] Y. Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.* 318(1-2):163-180, 2004.
- [Lei94] D. Leivant. Predicative recurrence and computational complexity I: word recurrence and poly-time. *Feasible Mathematics II*, pp. 320-343, Birkhauser, 1994.
- [LM93] D. Leivant, J.-Y. Marion: Lambda Calculus Characterizations of Poly-Time. *Fundam. Inform.* 19(1/2): 167-184, 1993.
- [LTdF06] O. Laurent, L. Tortora de Falco. Obsessional cliques: a semantic characterization of bounded time complexity. In *proceeding of LICS'06. IEEE 2006.*
- [Maz05] D. Mazza. Multiport Interaction Nets and Concurrency. In *Proceedings of CONCUR'05, LNCS 3653. Springer 2005.*
- [Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
- [MM00] J.-Y. Marion and J.-Y. Moyen. Efficient First Order Functional Program Interpreter with Time Bound Certifications. In *Proceedings of LPAR'00, LNAI 1955. Springer 2000.*
- [MO00] A. Murawski, C.-H. L. Ong. Discreet Games, Light Affine Logic and PTIME Computation. In *Proceedings of CSL'00, LNCS 1862. Springer 2000.*
- [MP07] J.-Y. Marion, R. Pechoux. Resource control of object-oriented programs. Presented at LCC'07 Lics affiliated workshop, july 2007.
- [MPW92] Robin Milner, Joachim Parrow, and David Walker, A Calculus of Mobile Processes, Parts I and II, *Information and Computation*, 100(1): 1-40 and 41-77, 1992.
- [MR06] V. Mogbil and V. Rahli. Uniform Circuits, & Boolean Proof Nets. In *proceedings of LFCS'07, LNCS 4514. Springer 2007.*
- [Nee04] P. Neergaard. A Functional Language for Logarithmic Space. In *Proceedings of APLAS'04, LNCS 3302. Springer 2004.*
- [NW06] K.-H. Niggl, H. Wunderlich. Certifying Polynomial Time and Linear/Polynomial Space for Imperative Programs. *SIAM J. Comput.* 35(5): 1122-1147, 2006.
- [Sch07] U. Schopp. Stratified Bounded Affine Logic for Logarithmic Space. In *Proceedings of LICS'07.. IEEE 2007.*
- [TZH02] D. Teller, P. Zimmer and D. Hirschhoff. Using ambients to control resources. *International Journal of Information Security* 2(3-4):126-144, 2004.
- [Ter04] K. Terui. Proof Nets and Boolean Circuits, In *Proceedings of LICS'04, IEEE 2004.*
- [Ter04b] K. Terui: Light Affine Set Theory: A Naive Set Theory of Polynomial Time. *Studia Logica* 77(1): 9-40, 2004.
- [WZN04] G. Winskel and F. Zappa Nardelli. New-HOPLA: A Higher-order Process Language with Name Generation. In *Proceedings of IFIP TCS'04, Kluwer 2004.*
- [YBH04] N. Yoshida, M. Berger, and K. Honda. Strong Normalisation in the Pi-Calculus. *Information and Computation*, 191(2):145-202, 2004.