

**Action Concertée Incitative**  
**NOUVELLES INTERFACES DES MATHEMATIQUES**

**I - FICHE D'IDENTITÉ DU PROJET**

**Nom du Projet :** (*maximum 20 caractères*)

GÉOCAL

**Titre du Projet :** (*maximum 3 lignes*)

Géométrie du calcul.

**Type du Projet<sup>1</sup> :**

"Petit" Projet	"Gros" Projet
	X

**Durée du projet<sup>2</sup> :**

**Description courte du Projet :** (*une demi-page maximum*)

Le but de ce projet est de favoriser la communication entre des équipes de mathématiciens (logiciens et topologues algébristes) et d'informaticiens, partageant des intérêts convergents dans le domaine de l'étude théorique du calcul. On espère promouvoir ainsi de nouvelles interfaces entre logique et topologie d'une part, et théorie de la concurrence, de la complexité algorithmique et de la programmation logique de l'autre.

**Coordinateur du projet :**

Nom	Prénom	Laboratoire (sigle éventuel et nom complet)
Ehrhard	Thomas	IML, Institut de Mathématiques de Luminy

**Organisme de rattachement financier pour le présent projet :**

CNRS

<sup>1</sup>Cocher la case correspondante au type du projet soumis.

<sup>2</sup>La durée d'un projet ne peut excéder 36 mois. Des demandes de projets d'une durée plus courte devront être particulièrement argumentées.

**Equipes ou laboratoires partenaires (nom complet et éventuellement sigle)<sup>3</sup>**

Institut de Mathématiques de Luminy (IML)
Preuves, Programmes, Systèmes (PPS)
Laboratoire d'Informatique Fondamentale (LIF)
INRIA-Sophia, projet MIMOSA
Unité de recherche FUTURS (INRIA)
Laboratoire de l'Informatique du Parallélisme (LIP)
Laboratoire d'Informatique de Paris Nord (LIPN)
LORIA, projet CALLIGRAMME
Institut de Mathématiques et Modélisation de Montpellier (I3M)
Laboratoire Spécification et Vérification (LSV)

---

<sup>3</sup>Insérer autant de lignes que nécessaire.

## Action Concertée Incitative

# NOUVELLES INTERFACES DES MATHÉMATIQUES

## B - DESCRIPTION DU PROJET

### B1 – Objectifs et contexte :

*On précisera, en particulier, les verrous scientifiques à dépasser, l'état de l'art ainsi que les projets concurrents ou similaires connus dans le contexte national et international, en particulier ceux auxquels les équipes du projet participent.*

Ce projet est principalement motivé par l'apparition récente de nouvelles formes d'interactions entre logique mathématique et informatique théorique, que l'on voudrait favoriser et développer.

Les liens entre mathématiques et informatique théorique sont innombrables, et il est donc essentiel de commencer par préciser de quelles mathématiques et de quelle informatique nous voulons parler ici.

Du côté informatique, on s'intéresse à l'informatique que l'on appellera « fondamentale » ; on désigne ainsi cette partie de l'informatique théorique qui vise à construire des outils abstraits pour étudier les langages de programmation, et développer de nouveaux modèles de calcul et de machines. Un point à nos yeux important relevant de l'informatique fondamentale, est l'étude de l'algorithmique parallèle, des protocoles réseau et des systèmes répartis (thèmes que l'on regroupe sous l'anglicisme « concurrence »). Cette problématique difficile, mais très en phase avec l'actualité technologique, a pris une importance primordiale au cours des dix dernières années.

Du côté des mathématiques, il s'agit pour l'essentiel (avec de notables exceptions que nous mentionnerons en temps utile), de deux branches qui ont commencé à se développer au début du vingtième siècle : *théorie de la récursivité* et *théorie de la démonstration*. Les préoccupations de ces deux disciplines étaient assez proches dès le début puisqu'il s'agissait de mathématiser (avec des motivations fondationnelles qu'il n'est pas nécessaire d'évoquer ici) deux activités jusqu'alors (à peu près) vierges de toute théorisation : calcul et démonstration.

Ces deux domaines ont connu un développement stupéfiant grâce à l'informatique, par exemple :

- la théorie de la récursivité a donné naissance à la théorie de la complexité algorithmique,
- des liens très profonds entre programmes et démonstrations sont apparus (isomorphisme de Curry-Howard) qui ont ouvert tout un champ d'applications de la théorie de la démonstration à l'informatique.

Si ces interactions entre ces branches des mathématiques et l'informatique sont si anciennes, quelle est la pertinence d'un projet ACI « Nouvelles interfaces des mathématiques » dans ce domaine ?

C'est que après une période d'intense collaboration dans les années 80-90 (notamment entre l'équipe de logique de Paris 7 et l'INRIA Rocquencourt) autour de l'isomorphisme de Curry-Howard et qui a produit de nombreux développements aussi bien théoriques qu'appliqués (programmation par preuve, sémantique dénotationnelle, logique linéaire, CAML, Coq, ...), les deux disciplines ont progressé dans leur voie propre. On a vu que l'informatique fondamentale s'intéresse (entre autre) à l'étude des systèmes concurrents, et plus récemment à des problématiques liées aux questions de sécurité sur les réseaux. De son côté la logique a continué à développer des outils abstraits d'analyse des langages de programmation : sémantique dénotationnelle, sémantique des jeux, étendant au passage leur domaine d'application à des langages de plus en plus riches.

Si l'on compare la situation aujourd'hui, à celle d'il y a 10 ans, à la notable exception de l'équipe PPS qui mélange des chercheurs des deux bords, il est de fait que les liens entre mathématiciens et informaticiens français sur les thèmes qui nous intéressent ici (informatique fondamentale et logique mathématique) se sont un peu distendus. Or sur le plan scientifique, il y a tout lieu de penser que la confrontation des concepts et outils modernes qu'elles ont développés indépendamment, serait extrêmement fructueuse. Par exemple, les calculs de processus occupent aujourd'hui beaucoup d'informaticiens théoriciens, alors que ce sujet est encore presque ignoré des logiciens, mais, comme on le détaille plus loin, la situation évolue rapidement.

**École d'hiver conclusive.** L'un des objectifs du projet est l'organisation à terme d'une école d'hiver, qui sera l'occasion de se livrer à un bilan scientifique des acquis. Celle-ci aura donc lieu durant l'hiver 2006. Dans la mesure du possible, on tâchera de la synchroniser avec l'un des « trimestres thématiques » que l'IML organise chaque année au CIRM sur le campus de Luminy<sup>4</sup>. L'école s'adressera à un public

<sup>4</sup>voir <http://iml.univ-mrs.fr/> li2002 pour la description du trimestre 2002 organisé par l'équipe de logique de l'IML

d'étudiants de 3ème cycle et de postdocs et devrait proposer plusieurs cours sur les thèmes majeurs du projet, mais pourra également être complétée par un colloque plus spécialisé.

Après cette brève introduction nous allons maintenant relever un certain nombre de thèmes appartenant à chacune des deux disciplines en dégagant les liens existant entre les points de vue informatique et logique.

## B2 – Description du projet : (5 à 10 pages)

*Entre autres, seront précisés le caractère innovant du projet, la valeur par la coopération et l'intérêt du projet pour le champ d'applications concerné.*

### Sémantique

L'un des domaines où les liens entre théorie de la démonstration et informatique théorique sont apparus le plus tôt est celui de la sémantique des langages de programmation. Cela résulte de l'isomorphisme de Curry-Howard qui est au départ la constatation, interne à la logique, d'une correspondance entre le  $\lambda$ -calcul d'une part (qui est l'un des formalismes introduits dans les années 30 pour définir la notion de calculabilité) et la logique intuitionniste de l'autre. La correspondance est très forte et identifie types des  $\lambda$ -termes et formules,  $\lambda$ -termes et preuves,  $\beta$ -réduction et élimination des coupures. Le  $\lambda$ -calcul étant ensuite apparu comme un outil très efficace en informatique, pour définir et implémenter certains langages de programmation dits « fonctionnels » (LISP en étant le premier exemple), et aussi comme un outil pour spécifier formellement l'aspect opérationnel des langages de programmation (comment chaque instruction doit se comporter), un lien très profond entre logique et informatique était du même coup établi. La *sémantique dénotationnelle* qui vise à interpréter les programmes comme des éléments de certains espaces associés à leurs types (classiquement, ces espaces sont des ensembles partiellement ordonnés, mais de nouvelles interprétations de nature très différente sont apparues récemment) est ainsi devenue à part entière une branche de la théorie de la démonstration, qui l'utilise et la développe pour interpréter les preuves.

L'utilisation et l'approfondissement de cette connexion programmes/preuves ont été très actifs en France dans les années 70-90, et le restent, même si l'excitation initiale a inévitablement un peu baissé. Nous détaillons ci-dessous quelques sous-thèmes de ce vaste sujet.

**Logique linéaire.** La logique linéaire tire ses origines d'une analyse de la sémantique dénotationnelle du système F (logique intuitionniste du second ordre, mais aussi langage de programmation à types polymorphes). Elle donne un statut logique aux règles structurelles (contraction et affaiblissement qui permettent de ne pas utiliser ou d'utiliser plusieurs fois une hypothèse) et fait apparaître une duplication des connecteurs usuels; conjonction et disjonction apparaissent en deux versions, multiplicative et additive. Elle a renouvelé la théorie de la démonstration en introduisant le concept de *réseau de preuves* (nouvelle représentation plus canonique des démonstrations), la *géométrie de l'interaction* qui rend compte algébriquement de la réduction des réseaux, et a révolutionné la sémantique dénotationnelle en mettant la *dualité* au premier plan (au sens de l'algèbre linéaire).

*Perspectives :* la possibilité apparue récemment de construire des modèles de la logique linéaire au moyen d'espaces vectoriels (de dimension infinie) et de fonctions linéaires continues a permis d'étendre le  $\lambda$ -calcul par des opérateurs différentiels. Les premiers résultats obtenus montrent que dériver un  $\lambda$ -terme n'est pas un pur jeu formel mais a un sens opérationnel précis, en rapport avec les machines à environnement et la réduction linéaire de tête. Certains aspects des calculs obtenus (non déterminisme, et présence d'un analogue de la « composition parallèle ») laissent entrevoir un lien avec les calculs de processus que nous voulons explorer dans ce projet. Nous souhaitons ici promouvoir l'interaction entre IML, LIF et LIP.

Comme on le verra un peu plus loin (section « Complexité algorithmique et élimination des coupures »), le paradigme de Curry-Howard a été étendu aux calculs à complexité bornée. En particulier, des modèles dénotationnels correspondant à ces systèmes ont été découverts et ainsi, un nouveau domaine de recherche a été ouvert, dont de nombreux aspects restent à explorer. Existe-t-il des interprétations faisant l'économie de la « stratification » utilisée dans ces modèles? Peut-on obtenir des résultats de complétude dénotationnelle? Ce thème devrait bénéficier de l'interaction PPS, IML et LIPN.

**Jeux et ludique.** Les interprétations *interactives* se sont imposées. Elles proposent un autre point de vue sur la dynamique des preuves et des programmes en remplaçant la réécriture ( $\beta$ -réduction, élimination des coupures) par une description des dialogues possibles entre une preuve et une contre-preuve, ou entre un programme et son environnement. Les modèles de jeux, mais aussi plus récemment la ludique,

relèvent de cette approche. Outre des modèles « complets » (c'est-à-dire dans lesquels tout élément est l'interprétation d'une preuve ou d'un programme) des langages fonctionnels et des preuves, cette approche fournit de nouveaux concepts pour rendre compte de certaines réalités informatiques dont l'approche fonctionnelle ne rend pas compte (états, pointeurs, par exemple).

*Perspectives* : l'importance croissante du calcul quantique dans la recherche en informatique théorique suscite de nouvelles perspectives, notamment après le travail de Peter Selinger (université d'Ottawa) qui propose un langage de programmation pour le calcul quantique et une sémantique à base d'opérateurs pour ce langage. Par ailleurs, Girard a proposé une version quantique des espaces cohérents (la sémantique dénotationnelle « canonique » de la logique linéaire), et plus généralement, il semble que le moment soit venu de reconsidérer d'un point de vue quantique l'ensemble de notre cadre de travail (et tout particulièrement l'approche interactive). Ce sera un chantier très prospectif de notre projet, qui sera mené essentiellement par IML, LIPN, PPS et FUTURS.

D'autre part, des travaux récents de James Laird (université de Brighton) ouvrent la voie à une nouvelle approche « extensionnelle » des modèles de jeux, dans laquelle, bien que représentant des stratégies déterministes, les morphismes sont des *fonctions* caractérisées par des propriétés de préservation inédites. Ces travaux ont suscité un fort intérêt, notamment en France (PPS et IML), et nous pensons qu'ils permettront une reformulation plus conceptuelle des modèles de jeux et de la ludique, avec des retombées certaines mais difficilement prévisibles sur la sémantique des langages de programmation. Ce thème restera probablement localisé à PPS et IML, du moins dans un premier temps.

**Logique classique et polarités.** Initialement restreint à la logique intuitionniste (c'est-à-dire, sans la règle du tiers-exclu), le paradigme de Curry-Howard a été étendu à la logique classique, d'une part sous l'impulsion des informaticiens qui se sont rendu compte que le `call/cc`, instruction de contrôle du langage SCHEME (de la famille LISP), se type naturellement au moyen d'une tautologie *classique*, et d'autre part dans le cadre de la logique linéaire. De nouveaux systèmes de logique classique ont ainsi vu le jour, qui, grâce notamment à la notion de formule *polarisée* et aux propriétés associées de *focalisation*, pallient le défaut de déterminisme du calcul des séquents classique de Gentzen (logique polarisée). Comme la ludique, ces travaux ont mis en évidence l'importance des polarités, non seulement pour la « détermination » de la logique classique, mais aussi pour les interprétations interactives (jeux). *Perspectives* : il conviendrait tout d'abord d'étudier l'extension de ces systèmes au second ordre (types polymorphes), qui est la clef pour la production de langages de programmation expressifs basés sur ces systèmes classiques. Une autre tâche sera de continuer l'exploration des modèles dénotationnels de ces systèmes classiques, et aussi de continuer l'exploration de la dualité apparue récemment entre les deux classes de traductions des systèmes classiques dans la logique intuitionniste : traductions « par passage de continuation », par nom d'une part et par valeur de l'autre. C'est un domaine qui fait déjà l'objet d'une coopération entre IML et PPS.

D'autre part, le caractère polarisé des systèmes de types utilisés pour typer certains sous-ensembles du  $\pi$ -calcul suggère un autre lien possible entre calculs de processus, dont l'analyse pourrait passer par l'étude des traductions du  $\lambda$ -calcul dans le  $\pi$ -calcul, qui sont des traductions par passage de continuation, tout comme les traductions des systèmes de logique classique dans le  $\lambda$ -calcul. Ce chantier plus prospectif devrait engager pour l'essentiel LIF, LIP et PPS.

**Machines.** Les machines abstraites à environnement, initialement conçues comme des modèles pour la compilation des langages fonctionnels, sont devenues des outils d'analyse de la  $\beta$ -réduction et de ses extensions classiques. De nouvelles notions de réduction plus fines ont ainsi été introduites (comme la réduction linéaire de tête, mais aussi le calcul sur les arbres de Böhm), qui ont des rapports profonds avec les interprétations interactives mentionnées ci-dessus, jeux et géométrie de l'interaction en particulier.

*Perspectives* : le  $\lambda$ -calcul différentiel pourrait également être mis à profit ici. Il permet en effet d'effectuer le développement de Taylor d'un  $\lambda$ -terme standard, et les termes de ce développement (qui sont en général en quantité infinie) rendent compte du comportement de ce terme dans la « machine de Krivine » qui est une de ces machines abstraites à environnement. Cela ouvre des perspectives de recherche sur les liens entre machines,  $\lambda$ -calcul différentiel, jeux et géométrie de l'interaction. Ce thème concernera principalement IML, LIPN et PPS.

**Réalisabilité.** La réductibilité et la réalisabilité sont des techniques standard de théorie de la démonstration permettant par exemple de prouver des théorèmes de normalisation (terminaison de la procédure d'élimination des coupures), ou de modéliser divers systèmes de logique intuitionniste. Elles ont été mises à profit en informatique théorique à de nombreuses fins et ont notamment permis de clarifier ce que signifie pour un programme de satisfaire une spécification. Plus récemment, Jean-Louis Krivine a initié un programme de recherche visant à extraire la signification algorithmique de théorèmes ou d'axiomes classiques de mathématiques (pour l'instant, surtout des résultats de théorie des ensembles et de logique)

par des techniques de réalisabilité, sous la forme de termes de  $\lambda$ -calcul étendus par des opérateurs élémentaires munis de règles de réduction adaptées au problème considéré. Pour ne citer qu'un exemple, il associe à l'axiome du choix dénombrable un opérateur simulant une « horloge » (au sens des machines). *Perspectives* : jusqu'à présent, les extensions du  $\lambda$ -calcul nécessitées par ce type d'extraction d'algorithmes ont toujours été de nature séquentielle. Il serait très intéressant de trouver des résultats mathématiques dont le contenu algorithmique s'exprime en termes de calcul concurrent, et ce sera une tâche qui devrait intéresser PPS, IML, LIF et LIP.

**Sémantiques dénotationnelle et théorie des probabilités.** Les études menées depuis 1998 par Panagaden, Desharnais et leurs collaborateurs ont permis de décrire les automates probabilistes interactifs de manière très générale en empruntant au vocabulaire et aux concepts principaux de la théorie des probabilités.

Un des bénéfices de cette approche est d'avoir des fondements clairs pour l'étude des systèmes interactifs probabilistes. Cette approche permet également de simplifier et de mieux comprendre les notions d'équivalence de processus dégagées de manière empirique dans le cas fini et de d'espérer une étude fine des approximations de processus qui mêle les concepts traditionnels de la théorie des probabilités comme l'espérance conditionnelle et la théorie des martingales, à des concepts de l'informatique théorique issus de la théorie des domaines.

*Perspectives* : Au-delà des ces premières applications on peut deviner un lien profond entre les notions traditionnelles de la sémantique dénotationnelle, développées dans le but d'étudier l'équivalence de programme, et la théorie des processus aléatoires qui s'est révélée féconde, y compris dans l'étude des systèmes différentiels déterministes. Nous nous proposons de développer plus avant les relations entre ces deux théories.

## Calculs de processus et logique

Les *calculs de processus* sont des formalismes permettant de modéliser le comportement de systèmes concurrents. Ces calculs proposent des opérateurs permettant à un ensemble de processus (programmes) s'exécutant indépendamment de se synchroniser, éventuellement pour échanger de l'information. Originellement motivés par l'étude des programmes concurrents, l'importance de ces calculs vient maintenant surtout du fait qu'ils fournissent des outils de modélisation, et donc de vérification, des protocoles mis en œuvre sur les réseaux informatiques.

Dans les calculs de processus, la *mobilité* a pris une part très importante. Il s'agit de rendre compte du fait qu'un processus peut migrer d'un environnement à un autre, par exemple, les « applets » du Web, les « portables » dans les réseaux de téléphonie mobile, etc. On a vu également récemment des applications de ces idées à la modélisation de systèmes biologiques.

Les théories de la mobilité (la première d'entre elles a été le  $\pi$ -calcul, qui a ensuite inspiré d'autres formalismes souvent plus adaptés aux situations concrètes, comme le calcul des *ambiants* et le *join-calculus*) sont fondés sur l'usage des *noms*, qui peuvent identifier des canaux de communications ( $\pi$ -calcul) ou des processus tout entiers (ambiants). L'essentiel est que dans ces formalismes, tout processus est en quelque sorte « localisé » en une place déterminée par les noms qui permettent de communiquer avec lui, et que cette localisation peut changer au cours du temps ; c'est l'idée même de la mobilité. Ces calculs ont donné lieu à de très nombreux travaux théoriques visant à déterminer leur pouvoir expressif (typiquement, le  $\lambda$ -calcul pur est représentable dans le  $\pi$ -calcul, ce qui montre que son mécanisme de passage de noms est extrêmement puissant du point de vue calculatoire), à introduire des notions d'équivalence entre processus (bisimulations), à donner aux processus des types donnant des informations sur leur comportement (par exemple, absence de deadlock), etc.

Les applications de ces formalismes sont nombreuses, en particulier, ils sont très employés pour spécifier des protocoles de communication et les protocoles cryptographique, et démontrer leur correction (et leur inviolabilité par des agents mal intentionnés).

Il se trouve que depuis quelques années, une idée de localisation rappelant beaucoup celle que nous venons d'évoquer a fait son entrée en force en logique : on en trouve les prémices dans la *géométrie de l'interaction* et dans les jeux, et elle s'exprime pleinement en *logique linéaire indexée* et surtout en *ludique*. L'objet de base de la logique est la formule, mais en théorie de la démonstration, on passe son temps à parler d'*occurrence* de formule, et non de formule, car le mécanisme de calcul fondamental (l'*élimination des coupures*) porte sur les occurrences. La ludique prend cette observation au sérieux à tel point que les formules disparaissent en tant qu'objet de base et sont remplacées par des « lieux » qui ne sont pas autre chose que les noms que les agents de base doivent partager pour pouvoir interagir (au sens de l'élimination des coupures). Il en est résulté un changement de point de vue mettant les formules au second rang : elles deviennent des collections de *desseins*. Ces idées de localisation en logique ont permis de mettre au point une version du  $\lambda$ -calcul travaillant sur des « records », au sens des langages de programmation objets.

L'un des axes prioritaires de notre projet sera donc d'explorer les liens entre ces deux instances de l'idée de localisation, la première de nature nettement informatique, et la seconde émanant de la théorie de la démonstration. Nous en espérons un renouvellement du paradigme de Curry-Howard qui, idéalement, donnerait de nouveaux outils logiques pour l'étude des systèmes concurrents et mobiles. Cet axe concernera essentiellement le LIF, le LIP, l'IML, PPS et le LIPN, mais nous espérons aussi sur ce sujet des interactions fécondes avec le LSV, l'IRMA et le GTA (le groupe Géométrie, Topologie et Algèbre de l'ISM).

Il faut mentionner aussi, dans le domaine de l'étude des calculs de processus et de la mobilité, l'existence de systèmes de logique dites « spatiales » qui, en plus de connecteurs usuels, mettent en jeu les noms, et donc l'aspect locatif des processus. L'un des objectifs de notre projet sera la compréhension des liens que ces systèmes logiques entretiennent avec les systèmes logiques locatifs, comme la logique linéaire indexée. Cette étude concerne principalement le LIP, le LIF, l'IML et PPS.

Nous souhaitons développer en France une direction de recherche récemment initiée par Glynn Winskel (université de Cambridge) qui utilise une approche catégorique de la concurrence : dans celle-ci, les processus sont vus comme des pré-faisceaux d'ensembles sur une catégorie de « chemins » représentant les formes possibles de traces d'exécution. Cette approche a l'avantage de rendre compte très naturellement du concept, central en parallélisme, de *bisimulation*, au moyen de la notion abstraite d'application ouverte entre préfaisceaux. Elle est exploitée pour mettre au point de nouveaux calculs de processus dont la sémantique est de ce fait directe (alors que l'interprétation des systèmes actuels est souvent acrobatique). Or les modèles en jeu se trouvent être précisément des modèles de la logique linéaire, et il semble donc naturel de mettre à profit ce lien pour comparer les langages de processus obtenus aux preuves de la logique linéaire. Nous voyons par conséquent dans cette approche un moyen, complémentaire du précédent, d'étendre au parallélisme le paradigme de Curry-Howard. Ce thème intéresse principalement le LIF, PPS et le LIPN, mais l'IML pourrait s'y mettre aussi, et le rôle de la notion de traces d'exécution (qui sont les objets de base dans l'application de la topologie algébrique au parallélisme) suggère une coopération novatrice avec le LSV, l'IRMA et le GTA.

## Parallélisme et topologie algébrique

Un autre thème que nous souhaitons intégrer à notre projet est celui des interactions entre topologie algébrique et parallélisme. Bien que les premiers travaux dans cette direction remontent à une dizaine d'années, c'est un sujet si actif, notamment en France, et riche de promesses, qu'il nous semble bien relever du programme « nouvelles interfaces ». L'idée essentielle consiste à voir un programme parallèle comme un « espace de flots » c'est-à-dire un espace topologique de configurations machine sur lequel on peut observer les traces d'exécution. Le non-déterminisme dû à l'exécution d'actions indépendantes crée de nombreuses traces possibles, toutes « équivalentes » au sens où quelque soit l'ordonnement choisi, le calcul effectué sera le même : les traces produites sont équivalentes modulo déformation (homotopie). Par contre, toute synchronisation crée des trous dans ces espaces, et donc, des inéquivalences.

Du coup, on retombe sur un vieux thème mathématique : la classification des espaces topologiques (modulo déformation), mais en fait, pas tout à fait, et il a fallu un certain temps pour le réaliser. Il y a une notion très importante dans le calcul, c'est l'irréversibilité du temps. La notion de déformation qu'il faut étudier doit donc respecter cette direction privilégiée. Cela donne lieu à de nombreux développements mathématiques nouveaux et profonds, alliant topologie, algèbre, logique, théorie des catégories etc. Des applications spécifiques de cette théorie naissante sont apparues récemment, par exemple pour l'analyse et la vérification de programmes concurrents (notamment pour le cas particulier des points morts et des états inatteignables) ; le CEA a d'ailleurs lancé une action en ce sens, avec un projet de développement d'analyseur statique, pour EDF, basé sur ces approches géométriques.

Les travaux théoriques dans ce domaine concernent principalement PPS, l'IRMA et le GTA, mais nous souhaitons que notre projet soit l'occasion pour cette communauté de développer davantage de liens avec les autres approches de la concurrence, à base de logique et de calculs de processus et donc nous encouragerons des coopérations entre les équipes mentionnées ci-dessus d'une part, et LIF, LIP et LIPN de l'autre.

Nous souhaitons également développer des connections entre cette approche géométrique de la concurrence et des approches géométriques qui ont été développées dans le domaine des systèmes de réécriture. Ici, un résultat de base est le théorème de Squier qui fait un lien entre l'existence d'un système de réécriture canonique fini, présentant un monoïde, et la finitude de son troisième groupe d'homologie (ceci ayant été généralisé ensuite à tous ses groupes d'homologie). Ces résultats ont été récemment étendus au sein du GTA à des systèmes de réécriture portant sur des théories algébriques plus complexes et on espère des liens, via notamment l'homotopie orientée, avec des travaux récents sur la géométrie de la réécriture, à PPS notamment.

**Logique, calcul et topologie algébrique.** La topologie algébrique a des interfaces non seulement avec la sémantique du parallélisme, mais aussi avec la logique. Des premiers travaux portant sur une extension d'un théorème de Friedman de complétude équationnelle de la logique S4 intuitionniste ont montré que S4 avait un contenu topologique. Plus précisément, pour démontrer l'analogie de ce théorème pour S4, qui essentiellement dit que deux programmes typés en S4 ont les mêmes valeurs si et seulement si ils sont syntaxiquement égaux (modulo conversion), il est nécessaire de démontrer l'existence d'une rétraction d'un ensemble simplicial sur un autre, un problème typique de topologie algébrique. De plus, on y utilise une structure dite de *sous-scone*, permettant de généraliser la notion usuelle en logique de *relation logique*, qui est de façon relativement frappante analogue à la notion de réalisation géométrique en topologie algébrique.

En général, l'étude des relations logiques et plus généralement des sous-scones pour des logiques telles que S4 ou la logique lax (correspondant au méta-langage de Moggi en informatique) mène à des structures intéressantes auxquelles on peut appliquer l'arsenal de techniques de topologie algébrique. La découverte récente que les sous-scones pour la logique lax dans une catégorie abélienne permettait de définir une théorie homologique des termes du méta-langage de Moggi en est un exemple typique.

Parmi les applications de ces techniques, on trouve notamment la définition d'un critère suffisant d'équivalence observationnelle pour des termes du méta-langage de Moggi, l'existence d'une relation logique reliant deux termes impliquant qu'ils sont observationnellement équivalents. L'équivalence observationnelle est une notion importante en informatique, étant notamment à la base de différentes notions de sécurité, parmi lesquelles la notion de non-interférence (impossibilité de faire dépendre des données publiques des valeurs de données sensibles), ou la notion de secret dans les protocoles cryptographiques codés en spi-calcul, une algèbre de processus due à M. Abadi et A. Gordon. On conjecture que l'utilisation de critères homologiques tels que ceux esquissés plus haut pourra permettre la découverte de nouveaux critères d'équivalence observationnelle, et partant de là, de sécurité.

On trouve aussi une connexion naturelle avec la logique linéaire, dont il a été question plus haut. La logique S4 ou la logique lax sont en effet en quelque sorte des abatardissements de la logique linéaire, où les règles structurelles de contraction et d'affaiblissement sont autorisées, contrairement à la logique linéaire. Les techniques de sous-scones semblent s'adapter naturellement à la logique linéaire. Ceci est peut-être la voie vers une sémantique homologique de la logique linéaire; en tous cas, il nous semble important de clarifier les relations entre logique linéaire et topologie algébrique, notamment via les constructions simpliciales et l'homologie.

## Complexité algorithmique et élimination des coupures

Le typage et la logique ont fourni les bases théoriques d'une part pour les langages fonctionnels comme CAML, qui autorisent une programmation sûre, et d'autre part pour les assistants de preuve comme Coq qui permettent de vérifier formellement la correction des programmes. Cependant ces outils fournissent peu de moyens pour étudier les aspects *quantitatifs* des programmes. Par exemple on peut extraire des preuves Coq par l'isomorphisme de Curry-Howard des programmes certifiés, mais on ne dispose pas de garantie sur l'efficacité de ces programmes.

L'évaluation de l'efficacité des programmes est traditionnellement le domaine de la complexité algorithmique, qui cherche à quantifier les ressources (temporelles et spatiales) nécessaires à l'exécution d'un algorithme. Or pour certaines applications en informatique on souhaiterait disposer de garanties certaines sur la complexité d'un programme *avant* son exécution : par exemple dans le cadre du code mobile un client peut vouloir s'assurer que le programme reçu d'un serveur s'exécutera avec un espace mémoire limité. Les méthodes formelles doivent donc être affutées pour être à même de rendre compte des propriétés de complexité de manière satisfaisante.

Or depuis quelques années de nouvelles approches pour contrôler la complexité issues de la théorie de la démonstration et de la théorie de la récursion ont vu le jour. Du côté de la théorie de la démonstration, la logique linéaire introduit une gestion explicite des ressources logiques (une preuve *linéaire* de  $B$  sous l'hypothèse  $A$  est contrainte à utiliser son hypothèse  $A$  une et une seule fois). On retrouve le pouvoir expressif habituel de la logique en introduisant des *modalités* logiques appelées *exponentielles* (mais cette terminologie n'a rien à voir avec la complexité) dont les règles, en termes d'élimination des coupures, peuvent être vues comme des instructions de gestion de l'espace (duplication et effacement). Il se trouve que, en restreignant ces règles, on parvient à définir des sous-systèmes de la logique linéaire (notamment la *logique linéaire légère* et la *logique linéaire douce*) dont l'élimination des coupures est en temps polynomial et qui de plus permettent de représenter *toutes* les fonctions (des entiers dans les entiers) pouvant s'exécuter en temps polynomial. Cette approche se situe clairement dans le paradigme de Curry-Howard : dans ces systèmes logiques, les preuves *sont* des programmes à complexité polynomiale.

Une autre approche relevant cette fois de la théorie de la récursivité a consisté à définir des langages de programmation abstraits basés sur une utilisation restreinte du schéma de récursion. Des disciplines plus ou moins libérales pour l'usage de la récursion permettent de caractériser ainsi plusieurs classes de

complexité. Le système de Bellantoni-Cook par exemple capture la complexité polynomiale en temps (PTIME) et d'autres formalismes ont été mis en avant notamment par Leivant et Marion. Les liens entre cette approche et celle de la logique linéaire ne sont pas encore complètement élucidés, même si des travaux de Murawski et Ong, puis Roversi ont jeté des passerelles entre les deux domaines.

Toutefois une des limites de ces deux branches de travaux concerne leur expressivité pratique. En effet même si toutes les *fonctions* PTIME sont programmables, ce n'est pas au moyen des *algorithmes* les plus naturels. Une autre approche plus pragmatique a été proposée par Hofmann, reposant sur un système de types linéaires (au sens de la logique linéaire) pour un langage fonctionnel. Le typage rend compte ici statiquement du nombre d'unités mémoires allouées par le programme. Il permet ainsi de garantir des bornes en espace et en temps. Cette approche permet de traiter de nombreux algorithmes usuels mais sa généralisation se heurte à des difficultés théoriques (par exemple pour l'extension à l'ordre supérieur, c'est-à-dire la manipulation de fonctions).

En résumé la théorie de la démonstration a trouvé dans la complexité un nouveau champ d'investigation, dans lequel les outils théoriques sont prometteurs et requièrent d'être confrontés aux défis de la pratique. Parmi les problèmes ouverts nous distinguons notamment les suivants.

- La mise au point de systèmes de typage permettant de garantir statiquement des bornes de complexité en espace et en temps sur les programmes, thème qui occuperait principalement CALLIGRAMME, LIF, LIPN et IML.
- Le développement de logiques pour raisonner sur les programmes et certifier qu'un programme est exécutable avec une certaine complexité, de la même manière qu'on certifie sa correction ou sa terminaison. L'approche de la logique linéaire légère pourrait être pertinente pour cette voie car elle permet de définir par exemple une théorie des ensembles PTIME. Ce thème concerne principalement LIP, IML et LIPN.
- La caractérisation d'autres classes de complexité que PTIME comme par exemple les classes de temps linéaire ou d'espace logarithmique. C'est un thème qui concerne prioritairement LIPN, LIF et IML.

## Programmation logique

À côté du paradigme de Curry-Howard, le paradigme de la programmation logique est une autre façon de concevoir les liens entre logique et calcul. Au lieu de correspondre aux preuves, les programmes sont maintenant de simples *formules* et, au lieu de l'élimination des coupures, l'exécution des programmes correspond à la *recherche* de preuve. Les multiples dynamiques possibles de calcul sont donc prises en compte par la multiplicité des combinaisons possibles de blocs logiques élémentaires (règles) pour construire une preuve. Cette approche a bénéficié des progrès réalisés ces vingt dernières années en logique, et en particulier de l'introduction de la logique linéaire, de la découverte de l'importance de la polarité des formules (focalisation) ainsi que des travaux sur les langages de termes d'ordre supérieur. Ces progrès permettent de voir la programmation logique comme une façon beaucoup plus efficace et naturelle de représenter les calculs et ouvre de nouvelles perspectives dans le domaine de la vérification formelle ou de la modélisation des calculs concurrents. Dans ce domaine, nous comptons renforcer les liens de IML (ludique, focalisation, construction de preuves partielles appliquées à l'ordonnancement) avec FUTURS (programmation logique en logique linéaire et applications).

Le paradigme de la programmation logique peut aussi être défini en modélisant les preuves par des réseaux. Dans ce cadre un programme est vu comme un ensemble de fragments de réseaux de preuves et l'exécution consiste à construire une preuve valide par assemblage de ces fragments. Ce point de vue, initié par Jean-Marc Andreoli, incite à réinvestir des concepts clés de la logique linéaire comme les critères de correction et la notion de module. Il permet en outre de modéliser naturellement le cadre de la programmation distribuée. D'autres aspects de la programmation comme la concurrence stricte ou la gestion d'un ordre sur les ressources pourraient aussi être pris en compte en revisitant le cadre logique sous-jacent. Ces directions concerneraient essentiellement l'IML et le LIPN.

Reste l'irritante insatisfaction d'avoir deux excellents paradigmes de calcul, chacun possédant de bons arguments pour prétendre à une certaine forme d'universalité, et néanmoins apparemment inconciliables. Et de fait les communautés de programmation logique et de langages fonctionnels entretiennent certes des liens, et particulièrement autour de la logique linéaire, mais peu de collaboration effectives. C'est l'un des mérites majeurs de la ludique que d'avoir enfin proposé un cadre qui rende compte à la fois de l'approche fondée sur l'élimination des coupures et de la recherche de preuves à travers l'idée de *construction interactive de dessin*. L'un des buts de notre projet sera donc d'approfondir ce lien récemment découvert entre le point de vue interactif sur les preuves (ludique, jeux) et la recherche de preuves. Sur ce chantier l'IML, le LIPN et FUTURS pourront coopérer.

### B3 – Résultats attendus :

L'exercice prédictif est toujours une gageure en mathématiques ; les progrès les plus spectaculaires se font dans des directions inattendues. Aussi il convient de prendre les spéculations suivantes avec précautions.

Toutefois, au vu des avancées récentes, on peut baliser les quelques thèmes qui seront traités prioritairement dans notre projet et pour lesquels des avancées sont « prévisibles ».

**Concurrence.** On l'aura compris, l'un de ces thèmes est la concurrence. La convergence déjà mentionnée entre les concepts de localisation en ludique et dans les calculs de processus, ainsi que les travaux les plus récents sur le typage des processus, laisse enfin entrevoir l'existence d'un lien fort entre théorie de la démonstration et concurrence. La mise à jour de ce lien devrait avoir des conséquences aussi importantes que celle de l'isomorphisme de Curry-Howard dans les années 50. Remarquons que notre projet rassemble la plupart des chercheurs français travaillant sur ce sujet, et réunit pour la première fois depuis longtemps le côté informatique théorique et le côté logique, si bien qu'il devrait rapidement se révéler très en pointe.

**Programmation logique.** Ce thème, lié au précédent, est également en train de connaître une évolution très rapide. L'unification qui s'annonce entre les deux modèles de calculs : programmation par preuve à la Curry-Howard et programmation logique, laisse présager d'importants bouleversements dans cette discipline. D'autre part ce thème offre de nombreux débouchés sur le plan applicatif, aussi bien du côté de la sécurité des réseaux, que de celui de l'architecture des applications réseaux (bases de donnée, serveur web).

**Topologie algébrique et calcul.** La topologie algébrique a fait apparition assez récemment dans le champ de l'informatique théorique et de la logique, avec la découverte qu'elle apportait des réponses naturelles aux problématiques liées à la correction des preuves en logique linéaire et qu'elle s'appliquait également dans le cadre de la théorie de la réécriture. Elle propose déjà un certain nombre de résultats tout à fait frappants. Le projet vise notamment à poursuivre l'étude du lien avec la concurrence, avec la perspective d'en extraire des solutions algorithmiques pour l'analyse des chemins d'exécution possibles dans les processus (deadlock, états inatteignables, ordonnancements « essentiels », etc.).

**Complexité.** On a vu qu'il existe deux grandes approches pour définir les fonctions polynomiales (s'exécutant en temps polynomial) : par la théorie de la récursivité ou par la logique. Or, même si des travaux reliant ces deux approches existent déjà, on comprends encore mal le lien les unissant ; notre projet se propose donc d'étudier ce lien plus à fond avec pour but l'unification de ces deux approches dans une théorie synthétique.