Principales formations d'initiation à la recherche de l'IRIF

Évaluation HCÉRES 2023

Période 2017-2022

Résumé

Ce document liste la liste des cours 2022-23 en masters LMFI et MPRI dont la responsabilité est assurée par un membre de l'IRIF. Il présente aussi l'EPIT que l'IRIF anime en collaboration avec son réseau de collaborations nationales depuis 1973.

Table des matières

1 Master LMFI

1.1 Overview

LMFI is the only French second year Master's programme dedicated to mathematical logic and its applications to computer science [1]. It trains high-level logicians and prepares them to later obtain a PhD, have an academic career, teach or work in research and development. It is jointly organized by two prestigious departments of the Université Paris Diderot and the CNRS, whose scientifc range cover most of mathematical logic and computer science : the Logic group of the Institut de Mathématiques de Jussieu and the Proof, Programs, Systems (PPS) pole of the Institut de Recherche en Informatique Fondamentale.

1.2 Curriculum

During the first term, LMFI consists of :
- an intensive introductory class in logic (30h, optional),
- four core classes (three 48h classes, one 84h class, 20 ECTS)
- core classes exercices session (36h each, 8 ECTS each) ;

and during the second term :
- a choice of eight advanced classes (48h each),

1. https://master.math.univ-paris-diderot.fr/en/annee/m2-lmfi/

– a choice of three ouverture classes (24h chacun),
– an research internship (Master's thesis), supervised by an academic.

Classes may be taught in English, if so requested by the students.

1.3 List of M2-courses coordinated by IRIF members in 2022-23

Proof Theory Thierry Joly (MCF UPCité)

– Completeness theorem of the LK sequent calculus with equality by Henkin's witnesses.
– Sequent calculus : cut elimination of and median sequent theorem in LK. Herbrand's theorem. LJ sub-calculation : intuitionist logic and its BHK interpretation. Properties of the sub-formula and existential witnesses in LJ.
– Natural deduction : NK and NJ systems. Cut elimination in NJ. Properties of the sub-formula and existential witnesses in NJ, then in HA (intuitionist arithmetic).
– Lambda-calculus : Confluence and standardization properties. Representation of recursive functions. T system. Curry-Howard correspondence. Realizability, strong standardization and program correctness.

Category Theory Francois Metayer (MCF Université Paris Nanterre)

The course presents the fundamental concepts of category theory, accompanied by numerous examples. The main goal is to pave the way towards the modern applications of category theory in logic, theoretical computer science and homotopy theory.

– Functors and natural transformations
– Universal constructions : limits and colimits
– Adjunctions and monads
– Brief introduction to higher categories

Functional programming and formal proofs in Coq Pierre Letouzey (MCF UPCité)

One half of this module will consist of course work, the other half will consist of practical work on a machine. The course will finish with a project to be carried out in Coq. The first part of this course is a prerequisite for the Homotopy Type Theory course.

Functional programming in Coq

– Lambda-calculus, recursivity (and its limitations in Coq), ML style typing
– Common data structures (Boolean, integers, lists, options, trees, etc.)
– Generic programming : polymorphism, modules, type classes
– Dependent types
– Connections with other languages such as OCaml, presentation of the "absent" from Coq (exceptions, imperative traits, input-output,...), how to make the link between OCaml and Coq

Formal proofs in Coq

– Specification language, basic logical rules
– Arithmetic, calculation, rewriting, recurrence
– Proof of properties of the usual data structures
– Inductive predicates
– Automation
– A quick overview of other proof assistants and comparison with Coq

Strategic balance of logic : games and models Mirna Dzamonja (Chercheuse Marie-Curie, CNRS)

The module will explore the ways that logic interact with the theory of games. Some examples where game theory enters set theory are strategic closure and determinacy, for model theory Ehrenfeucht-Fraïssé games, for descriptive complexity the pebble game, for automata theory certain decision arguments. Jouko Väänänen states that there are essentially three kinds of games in logic, and that there are essentially connected, forming a « strategic balance of logic ». We shall explore that balance. Lectures will be based on « Models and games » by Jouko Väänänen, with some additional explorations into descriptive set theory, large cardinals and decidability.

Proofs and programs : classical tools Antonio Bucciarelli (MCF UPCité) et Claudia Faggian (CR CNRS)

Proof theory has undergone at least two major developments over the past century as a result of Gödel's incompleteness theorems. The first took place in the 1930s, immediately after the results on incompleteness, with the introduction and study of natural deduction and sequent calculus s by Gentzen and lambda-calculus by Church. Church then showed the undecidability of predicate calculus via lambda-calculus while introducing a universal computation model while Gentzen deduced the consistency of various logical systems as a corollary of cut elimination of breaks in sequent calculus.

The second stage took place in the 1960s with the gradual highlighting, through the Curry-Howard correspondence, of the profound links between proofs and programs, from the correspondence between simply typed lambda-calculus and minimal propositional natural deduction to the various extensions of this correspondence to the second order, to classical logic and to the emergence of the notion of linearity in proof theory. Linear logic has profoundly renewed the links between the formal semantics of programming languages on one hand and proof theory on the other. Linear algebra is the third pole of this correspondence, focusing on the notion of computational resource.

The first part of the course focuses on the denotational semantics of functional programming, the second one on the operational semantics. Both build on proof theory and the theory of lambda-calculus.

First half :

- Interpretation of proofs and programs : interpretations of simply typed lambda-calculus and PCF in a closed Cartesian category (examples of CCC, adequacy theorem).
- Examples of models (Scott domains, coherence spaces, game models).
- Definability and full abstraction problems.
- Logical relations.

Second half :

- Tools to analyze the operational properties of a system : Abstract Rewriting Theory.
- Induction and Co-induction proof principles.
- Bridging between lambda-calculus and functional programming. Call-by-Value and Call-by Name, weak and lazy calculi. Big-Step and Small-Step operational semantics. Observational equivalence.
- Reasoning on programs equivalence . Bisimulation and coinductive methods.
- Linear Logic and Proof-Nets.
- Beyond pure : Probabilistic lambda-calculi.

Second order quantification and fixed-points in logic Alexis Saurin (CR CNRS) et Thomas Colcombet (DR CNRS)

This course will study different aspects of the notion of second order in logic and will study fixed-point logics. In particular, various applications of these logical formalisms to the study of formal languages on words and trees and to the study of programming languages will be discussed.

Part 1 : Second-order logic and arithmetic, system F

- Syntax of second-order logic, natural second-order deduction, particular theories and fragments (PA2, HA2, MSO, F, ...).
- Second-order models : full models and Henkin models, second-order completeness.
- Proofs and programs : extension of the Curry-Howard correspondence (second-order and polymorphism) and study of the system F (definition, representation of data types, normalization theorems (Girard reducibility candidates, realizability), representative functions in F).

Part 2 : Logic of the second monadic order

- Definitions, reminders and complements on the theory of automata.
- Decidability on words (Elgott-Büchi-Trakhtenbrot) ; links with automata and monoids.
- The first-order in the second-order (Schützenberger theorem).
- Decidability on tree-like structures (Courcelle's theorem) and structural bounds on decidability (Seese's theorem).
- The compositional method of Feferman-Vaught and Shelah.
- The theory of infinite words (Büchi, Safra).
- The theory of infinite trees and links with game theory (Rabin, Gurevitch and Harrington).

Part 3 : Smaller and larger fixed points in logic

- Some results on fixed points.

– Syntaxes and semantics of logics with smaller and larger fixed points (inductive definitions à la Martin-Löf, mu-calculus for classical, intuitionistic and modal logics).
– Deduction systems for fixed-point logics : induction à la Park, Omega rule, circular proofs.
– Relations with automata, with monadic second-order logic.
– Completeness of the mu-calculus and other results.

Homotopical algebra and higher categories Pierre-Louis Curien (DR émérite CNRS) Homotopy theory, which is devoted to the study of spaces up to deformation, has given rise to a branch of algebra called homotopical algebra, in which tools are developed for dealing with structures in which laws like associativity do not hold exactly like in classical algebra, but up to homotopy, these homotopies being themselves subject to coherences, etc.

Homotopy theory has also a logical side, in which types, proofs of equality, and proofs of equality of proofs of equality are interpreted as spaces, paths, and homotopies between paths, respectively. The notion of fibration, that plays an essential rôle in homotopy theory, is tightly related with substitution in dependent type theory. This interplay has led to a new version of type theory called homotopy type theory.

The course is a follow-up of that on category theory taught in the first semester, but can be followed by students who have already some basic background in category theory. We shall introduce the important notions underlying the subject : enriched categories, model categories, as well as different approaches to the definition of higher catégories, notably via simplicial sets. We shall also hint at connected subjects such as operads and ∞-operads, that also have arisen from topology. The lectures will partly follow the flow of exposition found in recent books (Categorical homotopy theory by Emily Riehl, The homotopy theory of $(\infty, 1)$-categories by Julia Bergner, From categories to homotopy theory by Birgit Richter, Higher categories and homotopical algebra by Denis-Charles Cisinski, Simplicial methods for higher categories by Simona Paoli, which all offer opportunities to the interested students for learning more), with an eye on the links with homotopy type theory.

2 Master MPRI

2.1 Overview

The MPRI is a research-oriented Master programme in computer science [2] run jointly by the following institutions : Université Paris Cité (which is the coordinating institution), ENS Paris Ulm, and Université Paris-Saclay (which includes ENS Paris-Saclay), Institut Polytechnique de Paris (which includes École Polytechnique and Telecom Paris). Moreover the Master has special ties with the following universities and research institutions : Sorbonne Université, CNRS, INRIA, and CEA.

Its purpose is to train future scientists through intensive exposure to contemporary research in computer science. It is open to students who have completed a French Licence or equivalent Bachelor's degree at a French or foreign college or university.

Most MPRI graduates continue on with a doctorate (Ph.D.), although some students successfully enter professional life after obtaining their Master's degree from MPRI. The MPRI + doctorate combination naturally leads to careers in academia (teaching-research positions at universities, full-time research positions at research institutions) or in industrial research and development.

Most of courses are either in english by default, or in english upon request.

From a formal point of view, the Master MPRI corresponds to :
– the specialization Recherche en Informatique (MIR) of the Master Informatique of Université de Paris.
– the specialization Algorithmique et Fondements de la Programmation of the Master Informatique of PSL University.
– the specialization Algorithmique et Fondements de la Programmation (AFP) of the Master Informatique of University Paris Saclay.
– the specialization MPRI of the Master Foundations of Computer Science of IP Paris.

2. https://wikimpri.dptinfo.ens-cachan.fr/

2.2 Curriculum

The aim of the MPRI is to offer students both core training in the fundamentals of computer science and more specialised training constitutive of a true introduction to research. One or two research internships complete this classical training curriculum.

In view of the great variety of cultural and educational backgrounds which the students taking the MPRI present, we let the students select their own customised two-year curriculum, i.e. not only the courses they wish to follow, but also the number and type of internships.

The second year (M2 year) begins with a first semester dedicated to specialisation by way of level 2 advanced courses. The second semester is devoted to a research internship within a laboratory, either in France or abroad.

In addition to the above-mentioned internships abroad, students may request, subject to approval by the MPRI's Studies Committee, to take all or part of the first or second year courses at a foreign partner university.

The specialisations available include :

- Algorithms
- Automata and Formal Languages
- Automated Deduction
- Combinatorics and Computer algebra
- Computability and Complexity
- Cryptography, Coding, and Security
- Logics and Semantics of Programs
- System Programming, Analysis, and Verification

2.3 List of M2-courses coordinated by IRIF members in 2022-23

Linear logic and logical paradigms of computation Delia Kesner (PR UPCité)

Une analyse fine des calculs de séquents classiques et intuitionnistes permet de concevoir des logiques plus adaptées aux problèmes de l'informatique et de dévélopper, grâce à la correspondance de Curry-Howard des formalismes intermédiaires entre le λ-calcul et les vrais langages de programmation.

Ce cours a pour but de donner une vision d'ensemble des motivations et des applications d'une de ces logiques, la Logique Linéaire, qui permet une analyse plus fine des processus de démonstration et de calcul, et d'introduire les notions de base des calculs intermédiaires les plus connus. On montrera dans le cours comment ces deux approches se rejoignent, à travers des interprétations calculatoires adaptées.

Ce cours dédie une attention toute particulière aux aspects syntaxiques et calculatoires des formalismes logiques.

Models of programming languages : domains, categories, games Paul-André Melliès (CR CNRS)

Le cours introduira aux méthodes développées par la sémantique dénotationnelle pour décrire mathématiquement les langages de programmation fonctionnels, organisés autour d'un noyau de lambda-calcul. En plus du lien avec la théorie des catégories, le cours traitera des deux piliers du sujet : sémantiques qualitatives et quantitatives issues des domaines de Scott et de la logique linéaire, ainsi que leur correspondance avec des propriétés opérationnelles des langages de programmation aussi bien déterministes que probabilistes.

Foundations of real-time and hybrid systems Eugène Asarin (PR UPCité)

Hybrid automata (HA), combining a discrete and a continuous behavior are a natural mathematical model of cyberphysical systems, explored since early 90s. They allow precise modeling of various phenomena. Unfortunately, most of exact verification problems are undecidable even for simple classes of HA. For this reason, two lines of research are extremely important : identification of decidable subclasses of HA and development of methods and tools for approximate verification. The most important subclass of HA with decidable verification problems is the class of timed automata TA. This class is capable to represent and analyze discrete behaviors in continuous time, which is well-adapted to modeling real-time systems, communication protocols, scheduling problems. Verification algorithms, based on results of Alur

and Dill, are implemented in several software tools, and used in practice. On the other hand, TA are a popular object of theoretical studies.

Algorithmic verification of programs Ahmed Bouajjani (PR UPCité)

Automatic program verification is an important and active research field, which poses many theoretical and practical scientific challenges. The goal of this course is to present the algorithmic approach for program verification, using analysis techniques based on automata, logic (decision procedures), computing the set of reachable states (which is potentially infinite), invariant synthesis, abstraction, and under-approximation techniques for detecting bugs. The course focuses on concurrent or distributed data structures, it includes a thorough presentation of the algorithms used in the implementations of these data structures, and the relevant correctness criteria.

Parameterized Algorithms Valia Mitsou (MCF UPCité)

Parameterized complexity is a branch of computational complexity offering us tools in order to attack NP-hard problems, but also in order to solve more efficiently problems belonging in P. In parameterized complexity, we perform a more refined analysis of the algorithmic complexity of a problem : the running time is analyzed as a function of the input size as well as of one or more parameters. When the parameter is expected to be small, an algorithm polynomial in the input size but exponential in the parameter(s) is expected to be tractable, since the combinatorial explosion is confined to the (small) parameter(s).

In the first part of the course, we will discuss about the theory of parameterized complexity and define the different complexity classes ; we will make an exposition of the main classical techniques that we use in the design of parameterized algorithms : kernelization, bounded search trees, iterative compression and color coding. We will also learn how to design lower-bounds and make the connection of parameterized and approximation algorithms (another branch of algorithms design for attacking NP-hardness). The second part of the course will be devoted to the study of probably the most successful and well-known structural parameter : the treewidth of a graph. We will first give a quick overview of the graph minor project that defined treewidth, essentially lead by Robertson and Seymour in a series of 23 articles published between 1993 and 2010. We will then see how to use this theory in order to design FPT algorithms. In particular, we will study the celebrated Courcelles Theorem and more advanced methods such as bidimentionality.

Randomness in Complexity Frédéric Magniez (DR CNRS)

The common thread of the course is the use of randomness in algorithms and complexity. This year, the course will cover a series of ideas and techniques at the boundary between combinatorial and continuous optimization. In particular we will focus on graph algorithms, interpreted through the prism of continuous methods.

Analysis of algorithms Vlady Ravelomanana (PR UPCité)

Ce cours présente des techniques de combinatoire appliquées à l'analyse d'algorithmes. L'accent est mis sur la combinatoire analytique, où interviennent des méthodes de combinatoire énumérative (par fonctions génératrices) conjuguées à des méthodes danalyse asymptotique. Cette approche permet lévaluation des principaux algorithmes et structures de données de linformatique. Le cours est étayé de nombreux exemples. Les outils présentés s'appliquent à l'analyse d'algorithme, mais également à l'étude d'objets aléatoires tels que les permutations, les partitions, les mots issus de languages rationnels ou algébriques, et les graphes.

Ce cours relativement mathématisé doit bien convenir à des étudiants ayant une formation mixte en mathématiques et informatique de bon niveau, comme il s'en rencontre à l'École polytechnique et dans les Écoles normales supérieures, ainsi que dans les parcours Math-Info des cursus universitaires. Il peut aussi servir de passerelle vers l'informatique, pour des étudiants dont la formation jusqu'à Bac+4 était principalement mathématique. Pour ceux qui ont suivi des filières plus informatiques, il propose un socle méthodologique solide mais demande un certain investissement.

Finite automata modelling Matthieu Picantin (MCF UPCité)

Les automates finis sont l'un des modèles les plus simples de machines qui calculent, le premier dans toute hiérarchie de machines de Turing plus ou moins contraintes. Cette simplicité en fait un objet

robuste, susceptible de nombreuses définitions équivalentes relevant de la théorie de la complexité, de l'algèbre non-commutative et de la logique.

La théorie classique des automates finis traite d'automates qui acceptent, ou n'acceptent pas, des mots. L'objectif de ce cours est de montrer comment introduire des notions sortant de ce cadre, et comment se servir de ces extensions dans différents contextes. Ainsi, nous verrons des machines à états finies permettant de calculer différentes sortes de fonctions ou des relations. Nous verrons également comment ce type d'outils sert dans l'analyse de structures infinies et en particulier des groupes.

Distributed Algorithms for Networks Pierre Fraigniaud (DR CNRS)

Distributed Computing is dedicated to the design and analysis of algorithms performed by a set of autonomous computing entities whose objective is the realization of a common task, in absence of any coordinator, like in, e.g., Internet, P2P systems, cloud computing, wireless networks, social networks, insect colonies, school of fish, etc., and any decentralized system in general.

This course focusses on distributed computing in networks, in which the computing entities have to cope with uncertainty caused by spatial constraints. The typical problems to be solved in this context are graph problems, such as coloring, construction of a maximal independent set (MIS), construction of a minimum-weight spanning tree (MST), etc.

The design and analysis of distributed algorithms in networks is tightly connected to the following fields : graph theory, deterministic and randomized algorithms, communication complexity, interactive proofs, etc. The course may even provide examples of connections with algebraic topology, zero-knowledge proofs, and quantum computing.

Shared-Memory Distributed Computing Carole Delporte (PR UPCité)

Distributed computing concerns designing and understanding algorithms for sets of independent computing units that have to communicate to coordinate their activities. The problem space of distributed computing is vast and it would be impossible to undertake an exhaustive study within a single course. Even a small-scale distributed system may expose an amazingly complex behavior that would make it very challenging to formally reason about. But we have to meet the challenge ! Due to inherent limitations of centralized computing, all computing systems nowadays is becoming distributed, ranging from Internet-scale services to multiprocessors. Therefore, understanding the principles of distribution and concurrency is indispensable in all aspects of designing and engineering modern computing systems. The main challenge here is to balance correctness of the system's executions with its availability and efficiency, in the presence of possible misbehavior of the system components and the environment (such as faults and asynchrony).

This course discusses how to design distributed algorithms, reason about their correctness, and derive matching complexity bounds. The primary focus of the module is on understanding of the foundations of distributed computing. This course focuses on the models in which computing units communicate through a shared memory.

Game theory techniques in computer science Wieslaw Zielonka (PR UPCité)

Following the publication in 1944 of the book Theory of Games and Economic Behavior by John von Neumann and Oskar Mergenstern, and the consequent work of John Nash in the early 1950s, game theory has been used mainly as a model for economic and social interactions.

However, since the early 1980s, game theory has become increasingly important in computer science and the aim of this course is to present various game models and applications of game theory in several areas of computer science : automata theory, logic, program verification, optimization, and reinforcement learning.

Symbolic dynamics Valérie Berthé (DR CNRS)

Ce cours s'adresse à des étudiants possédant une grande curiosité, car il fait appel simultanément à plusieurs domaines des mathématiques (combinatoire, topologie, théorie de la mesure, théorie ergodique, etc.).

Lobjet de ce cours est de présenter une introduction à la dynamique symbolique. Un système dynamique discret (cest-à-dire à temps discret) est défini comme laction dune application T agissant sur un espace X. Il s'agit alors détudier lévolution à long terme du système. La dynamique symbolique a

pour objet létude de systèmes dynamiques discrets composés de suites infinies de symboles dun alphabet fini. Ils apparaissent naturellement comme des codages de trajectoires de points dun système dynamique défini sur un espace X, a priori de nature continue, selon une partition finie donnée.

Advanced graph theory Reza Naserasr (CR CNRS)

Graphs are fundamental tools for most areas of computer science. The first goal is for all students attending to the course to learn tools and techniques on dealing with problems on graph theory. This would help in particular with stating your research problem using language of graph theory when needed. The second goal of the course is to offer a deeper look for those student who are persuaded to continue graph theory as a main research career.

Complexité des circuits Sylvain Perifel (MCF UPCité)

Le but de ce cours est d'étudier la complexité de langages ou de polynômes en termes de circuits booléens ou arithmétiques. L'accent sera mis sur la preuve de bornes inférieures. Des notions de complexité booléenne et une certaine aisance en algèbre (polynômes) sont requises.

Quantum information and applications Sophie Laplante (PR UPCité)

Each year computing machines become faster and faster, but they use still use at their base the same Newtonian physics. Feynman in 1982 already asked about the necessity of this restriction to classical physics. The idea behind quantum computation is to use quantum phenomena to solve tasks that conventional machines cannot achieve.

Historically the first result that showed the superiority of the quantum model was in cryptography. Bennett and Brassard in 1984 gave a first quantum protocol for perfectly secure key distribution. Such an unconditional security does not exist in the classical world.

At present many important concepts of theoretical computer science have been extended to quantum computation, from communication to algorithms and error correcting codes.

The aim of this course is to present the bases of several concepts about quantum computation. The emphasis will be on quantum algorithms and communication. We will describe the basics of Quantum Computation and its applications in algorithms, communication complexity and nonlocality.

3 Ecole de Printemps d'Informatique Théorique

3.1 Présentation

Initiée en 1973 par Maurice Nivat et ses collaborateurs, l'Ecole de Printemps d'Informatique Théorique (EPIT) organise chaque année une école intensive dans un domaine de l'informatique théorique. Le public visé est composé de doctorant·e·s et de chercheur·euse·s confirmé·e·s qui souhaitent se spécialiser dans le domaine considéré.

3.2 Fonctionnement

Comité de pilotage (Avec mention des membres IRIF)

- Sylvie Corteel (jusqu'à 2027, IRIF)
- Pierre-Alain Fouque (jusqu'à 2023)
- Aurélien Garivier (jusqu'à 2024)
- Michele Pagani (jusqu'à 2025, IRIF) [Président]
- Daniela Petrisan (jusqu'à 2027, IRIF)
- Marthe Bonamy (jusqu'à 2028)

Anciens membres (depuis 2002) : Jean-Eric Pin (jusquà 2009, IRIF), Daniel Krob (jusquà 2010, IRIF), Jean-Michel Muller (jusquà 2011), Roberto Amadio (jusqu'à 2012, IRIF), Pierre-Louis Curien (jusquà 2013, IRIF), Jean Mairesse (jusqu'à 2015, IRIF), Anca Muscholl (jusqu'à 2015), Frédéric Magniez (jusqu'à 2016, IRIF), Guillaume Hanrot (jusqu'à 2017), Jean Goubault-Larrecq (jusqu'à 2018), Delia Kesner (jusqu'à 2019, IRIF), Valérie Berthé (jusqu'à 2021, IRIF), Cristina Sirangelo (jusqu'à 2021, IRIF), Nicolas Trotignon (jusqu'à 2022).

Statuts

- Le comité de pilotage est constitué de 6 membres, nommé·e·s chacun pour un mandat de 6 ans, non renouvelable.
- Le comité de pilotage élit en son sein un·e président·e, pour une durée de 3 ans, non renouvelable.
- Tous les votes au sein du comité de pilotage se font à la majorité absolue. En cas dégalité, la voix du ou de la présidente est prépondérante.
- Le remplacement dun membre du comité de pilotage à la fin de son mandat ou sur démission en cours de mandat, se fait par cooptation et vote des membres restants.
- Lannée n, le comité de pilotage sélectionne une proposition de lieu et de président·e du comité de programme et dorganisation pour lEPIT de lannée n+2.
- Toute modification concernant le présent texte, et relative au comité de pilotage est soumise à laccord de la majorité de lensemble des membres du comité de pilotage.

Demander à organiser une EPIT Typiquement, une EPIT dure une semaine et a lieu au Printemps dans un site à lécart des centres-villes. Les candidats à lorganisation dune EPIT soumettent au comité de pilotage un petit document qui comprend :

- Le nom du ou de la présidente du comité dorganisation.
- Une brève description de la thématique scientifique de lécole et de ses objectifs.
- Un site et la période envisagés.
- Un comité dorganisation et un comité scientifique.
- Une liste de cours et denseignants envisagés avec un argumentaire sur la cohérence scientifique et pédagogique de la proposition.
- Des éléments préliminaires du budget.
- Il convient de prendre contact avec le comité de pilotage 24 mois avant la date envisagée ; les points 3-6 ci-dessus pourront être peaufinés dans les mois qui suivent la prise de contact.

3.3 Historique

Les noms des écoles sont cliquables à partir de 1998.

1. Langages algébriques, Bonascre (J.-P. Crestin et M. Nivat), 1973
2. Complexité des algorithmes, Ile de Berder (Ph. Flajolet), 1974
3. Monoïdes syntactiques, Vic-sur-Cere (J.-F. Perrot), 1975
4. Sémantique des langages de programmation, Molines-en-Queyras (M. Nivat), 1976
5. Séries formelles, Vieux-Boucau-les-Bains (J. Berstel), 1977
6. Lambda-calcul, La Châtre (B. Robinet), 1978
7. Théorie des codes, Jougne (D. Perrin), 1979
8. Parallélisme, Colleville (G. Roucairol), 1980
9. Langages algébriques, Murol (L. Boasson), 1981
10. Compilation, Barèges (L. Nolin), 1982
11. Algorithmique, Ile de Ré (M. Fontet), 1983
12. Automates et mots infinis, Le Mont Dore (D. Perrin) 1984
13. Logique combinatoire et lambda-calcul, Val d'Ajol, (G. Cousineau et P.-L. Curien), 1985
14. Réseaux d'automates, Argelès-sur-Mer (C. Choffrut), 1986
15. Linguistique et informatique, Ile d'Oléron (M. Gross et D. Perrin), 1987
16. Automates finis et applications, Ramatuelle (J.-E. Pin), 1988
17. Logique et informatique, Albi (B. Courcelle), 1989
18. Sémantique des systèmes de processus concurrents, La-Roche-Posay (I. Guessarian), 1990
19. Mathématiques et informatique théorique, Mejanes-Le-Cap (S. Grigorieff), 1991
20. Parallélisme, Les Sables d'Or les Pins (M. Cosnard et Y. Robert), 1992
21. Réécriture de termes, Font-Romeu (H. Comon et J.-P. Jouannaud), 1993

22. Programmation logique avec contraintes, Chatillon-sur-Seine (A. Podelski), 1994
23. Géometrie et topologie discrète, Super Lioran (J.-P. Reveilles et D. Richard), 1995
24. Automates cellulaires, Saissac (J. Mazoyer), 1996
25. Algorithmique, Longefoy (D. Krob et M. Morvan), 1997
26. Algèbre Max-Plus et applications en informatique et automatique, Ile de Noirmoutier (S. Gaubert, J.-J. Loiseau, J. Mairesse, J.-E. Pin), 1998
27. Codage et cryptographie, Batz-sur-mer (A. Canteaut, C. Carlet, P. Charpin, M. Girault, B. Vallee), 1999
28. Pavages du plan, Branville (B. Durand, M. Nivat, L. Vuillon), 2000
29. Arithmétique des ordinateurs, Prapoutel-Les-Sept-Laux (C. Frougny et J.-M. Muller), 2001
30. Sémantique des langages de programmation, Agay (P.-L. Curien, V. Padovani, J.-M. Rifflet), 2002
31. Algorithmique distribuée, Porquerolles (C. Delporte, H. Fauconnier, R. Guerraoui), 2003
32. Théorie de la concurrence et applications, Luminy (R. Amadio, P. Gastin, R. Morin, M. Zeitoun), 2004
33. Complexité algorithmique, Montagnac-les-truffes (P. Koiran, F. Magniez, N. Portier), 2005
34. Jeux en sémantique et vérification, Ile de Ré (P.-A. Mellies, A. Muscholl), 2006
35. Ordonnancement, Fréjus (F. Vivien), 2007
36. Apprentissage automatique, Porquerolles (F. Denis, L. Ralaivola), 2008
37. Preuves de sécurité calculatoires et symboliques, Barbizon (H. Comon), 2010
38. Jeux, Carcans-Maubuisson (I. Walukiewicz), 2011
39. Algorithmique probabiliste, Ile de Ré (I. Kerenidis, C. Mathieu, F. Magniez), 2012
40. Réseaux euclidiens : algorithmique et applications, Vercors (G. Hanrot, D. Stehlé), 2013
41. Algorithmique et bioinformatique, Ile d'Oléron (S. Vialette), 2014
42. Preuve mécanisée de programmes, Fréjus (Y. Régis-Gianas), 2015
43. Graphes et surfaces : Algorithmique, combinatoire et topologie, Luminy (E. Colin de Verdière, G. Schaeffer), 2016
44. Algorithmique distribuée, Île de Porquerolles (A. Milani), 2017
45. Vérification de programmes, Aussois (D. Baelde, C. Enea), 2018
46. Donnéees, logique et automates, CIRM Luminy (A. Gheerbrant, L. Libkin, L. Segoufin, P. Senellart, C. Sirangelo), 2019
47. Théorie des types homotopiques (M. Sozeau, N. Tabareau), 12-16 avril 2021 (EPIT 2020).
48. Informatique Quantique, CIRM Luminy (O. Fawzi, E. Kashefi, S. Perdrix), 24-28 mai 2021 (EPIT 2021).
49. Apprentissage automatique, CIRM Luminy (O. Cappé, A. Garivier, R. Gribonval, E. Kaufmann, C. Vernade), 23-27 mai 2022.