Complexity Winter School

Marseille - January 30 - February 3 - 2012

Monday January 30

10h45 : Opening.

11h-12h30 : Yiannis N. Moschovakis - *Relative complexity in arithmetic and algebra.*

12h30 : Lunch.

14h30-15h : Pierre Valarcher - *From intentional behavior to algorithmic completeness.*

15h-15h30 : Ugo Dal Lago - *Higher-order Interpretations and Program Complexity (joint work with Patrick Baillot).*

15h30 : Coffee break

16h-16h30 : Isabel Oitavem - *NP with tier 0 pointers.*

16h30-17h : Reinhard Kahle - *Computational Complexity and Applicative Theories (joint work with Isabel Oitavem).*

Tuesday January 31

9h-10h30 : Martin Hofmann - *Pure pointer programs (implicit computational complexity) with an abstract datatype of pointers.*

10h30 : Coffee break

11h-12h30 : Yiannis N. Moschovakis - *Relative complexity in arithmetic and algebra.*

12h30 : Lunch.

14h30-16h : Emmanuel Hainry - *Computable Analysis: Computability and complexity over the reals.*

16H : Coffee break

16h30-17h : Marco Solieri - *Deep into optimality Complexity and correctness of shared implementation of bounded logics (joint work with Stefano Guerrini and Thomas Leventis) .*

17h-17h30 : Matthieu Perrinel - *Bornes fortes pour la logique linéaire par niveau.*

Wednesday February 1

9h-10h30 : Martin Hofmann - *Pure pointer programs (implicit computational complexity) with an abstract datatype of pointers.*

10h30 : Coffee break

11h-12h30 : Neil Jones - *Alan Turing and 75 years of Research in Models of Computation .*

12h30 : Lunch.

Thursday February 2

9h-10h30 : Heribert Vollmer - *Circuit complexity.*

10h30 : Coffee break

11h-12h30 : Stefan Szeider - *Parameterized complexity.*

12h30 : Lunch.

14h30-16h : Virgile Mogbil - *Parallel Computation with Boolean Proof Nets.*

16h-16h30 : Clément Aubert - *Proof circuits and others parallel models of computation.*

16h30 : Coffee break

Friday February 2

9h-10h30 : Stefan Szeider - *Parameterized complexity.*

10h30 : Coffee break

11h-12h30 : Heribert Vollmer - *Circuit complexity.*

12h30 : Lunch.

14h-14h30 : Martin Lackner - *Fixed-Parameter Algorithms for Finding Minimal Models (joint work with Andreas Pfandler).*

14h30-15h : Johannes Schmidt - *On the Parameterized Complexity of Default Logic and Autoepistemic Logic (joint work with A. Meier, M. Thomas, and H. Vollmer).*

Abstracts of the Lectures

Martin Hofmann - *Pure pointer programs (implicit computational complexity) with an abstract datatype of pointers.*
Pointer programs are a model of structured computation within LOGSPACE (logarithmic space on a Turing machine). They capture the common description of LOGSPACE algorithms as programs that take as input some structured data, e.g. a graph, and that store in memory only a constant number of pointers to the input, e.g. to the graph nodes.

We define a formalised language for pointer programs (Purple) and show that some LOGSPACE algorithms can be programmed in Purple while others cannot (e.g. reachability in undirected graphs). This yields a better understanding of the structure of LOGSPACE and also sheds new light on finite model theory ; indeed, since formulas in deterministic transitive closure logic (DTC) can be evaluated in Purple it can be deduced that reachability in undirected graphs cannot be defined by a DTC formula which was hitherto unknown. This also, somewhat trivially, separates Purple from PTIME. In order to get a more meaningful such separation we would like to strengthen Purple while still remaining strictly below PTIME. Possible such extensions are nondeterminism, iterators in the style of Java, and various patterns of recursion patterns.

The course will give an overview of Purple and related systems and results, in particular Graph Automata and DTC logic and present some of the extensions to Purple that we currently investigate. This is joint work with Ulrich Schoepp and Ramyaa Ramyaa.

Yiannis N. Moschovakis - *Relative complexity in arithmetic and algebra.*
The main aim of these lectures is to show how ideas from the study of recursive programs and algorithms can be used to derive lower complexity bounds for mathematical problems which are robust (with respect to the choice of computation model) and plausibly "absolute", i.e., they restrict "all algorithms". An alternative name for them would be "Recursion and Complexity". There will be four, 40-minute lectures, as follows :

(1) Recursive (McCarthy) programs. Mostly well-known introductory material. One possibly novel idea is a new approach to the foundational problem of justifying the Church-Turing Thesis, which comes from examining the connection between recursion and computation.

(2) Uniform processes. A simple, axiomatic approach to the theory of algorithms from specified primitives in the style of "abstract model theory". Uniform processes capture the "uniformity" of algorithms—that they apply "the same procedure" to all inputs—but not their effectiveness. They carry natural complexity measures and support a simple "Homomorphism Test" which can be used to derive absolute lower bounds for algorithms from specified primitives. This is the main,

new material in these lectures.

(3) Lower bounds in arithmetic. Strong versions of results obtained jointly with Lou van den Dries, mostly about the complexity of coprimeness from various primitives and relative to various computation models.

(4) Lower bounds in algebra. Strong versions of results on "0-testing" for polynomials over various fields, mostly due to Peter Buergisser (with others) for algebraic decision trees.

Stefan Szeider - *Parameterized complexity.* -
. Parameterized Complexity is a new theoretical framework for the analysis and solution of hard computational problems. Virtually in every conceivable context we know more about the problem input than just its size in bytes. The key idea of parameterized complexity is to represent this additional information in terms of a parameter, and to study computational problems in a two-dimensional setting formed by the input size and the parameter. This setting gives rise to a new theory of algorithms and complexity that allows a more fine-grained complexity analysis than the classical one-dimensional setting. Central to the theory is the notion of fixed-parameter tractability, which relaxes the classical notion of tractability by admitting algorithms whose runtime is exponential, but only in terms of the parameter of the problem instance. In recent years, ideas from parameterized complexity theory have found their way into various areas of computer science, such as artificial intelligence, database theory, computational logic, computational social choice, computational geometry, and computational biology.

In the first part of this tutorial we will discuss algorithmic methods for establishing fixed-parameter tractability, including the method of bounded search trees, reductions to a problem kernel, and algorithmic meta theorems. In the second part we will discuss the main concepts of parameterized intractability, which are similar to the theory of NP-completeness, and allow to provide strong evidence that a parameterized problem is not fixed-parameter tractable.

Heribert Vollmer - *Circuit complexity.*
In these introductory lectures we will cover two topics from the area of circuit complexity : In the first lecture we will talk about arithmetic circuits of small depth. We will mainly concentrate on the classes $\#\mathrm{NC}^1$ and $\#\mathrm{GapNC}^1$ and show how they provide useful characterizations of counting complexity classes, how they capture the computational complexity of some problems in linear algebra, and how they shed new light on the relation between logarithmic depth circuits and logarithmic space Turing machines. In the second lecture we will turn to the area of proof complexity and use very small NC^0 circuit families as proof checkers. Alternatively one might say that we use NC^0 circuit families to enumerate languages (allowing repetitions). We will show how on the one hand even NP-complete languages can be enumerated in this way but on the other hand some very simple

languages lack this property.

Abstracts of the Invited talks

Emmanuel Hainry - *Computable Analysis: Computability and complexity over the reals.*
Computing over continuous domains, in particular on the reals is quite important for example to simulate physical, biological, mathematical phenomena. Various models and various machines for computing on such domaines exist but there is no such thing as a Church-Turing thesis for computing on the reals, some models staying in the Turing-complete class, some using reals to answer the halting problem. We will concentrate on one model : Recursive Analysis which is well accepted and has a long history as it was already present in Turing's 1936 paper and even hinted at by Borel in 1912.

We will present the recursive analysis model, including some explanations on why some choices were made and introduce fundamental results on computability in recursive analysis. We will also show recent results on the characterization of classes of computable real functions.

Then, we will enter the complexity field. What does complexity mean when the size of the input is infinite and the computation is not terminating ? We will answer this question, present basic tools to analyse the complexity of functions and a framework that allows us to translate characterizations of discrete complexity classes into characterizations of the analog real complexity class, and use it to give a characterization of the class of real functions computable in polynomial time

Neil Jones - *Alan Turing and 75 years of Research in Models of Computation .*
Alan Turing and 75 years of Research in Models of Computation By Neil D. Jones and Jakob Grue Simonsen.
From a programming perspective, Alan Turing's epochal 1936 paper on computable functions introduced several new concepts, including what is today known as self-interpreters and programs as data, and originated a great many now-common programming techniques.
We begin by reviewing Turing's contribution from a programming perspective ; and then systematise and mention some of the many ways that later developments in models of computation (MOCs) have interacted with computability theory and programming language research.
Next, we describe the "blob" MOC : a recent stored-program computational model without pointers. Novelties of the blob model : programs are truly first-class citizens, capable of being automatically executed, compiled or interpreted. The model

is Turing complete in a strong sense : a universal interpretation algorithm exists, able to run any program in a natural way and without arcane data encodings. The model appears closer to being physically realisable than earlier computation models. In part this owes to strong finiteness due to early binding ; and a strong adjacency property : the active instruction is always adjacent to the piece of data on which it operates.
Next, some of the best-known among the numerous existing MOCs are overviewed and classified by qualitative rather than quantitative features, paying attention to two factors of prime importance to programmability and physical realizability : finiteness (and with respect to what) ; binding times (of what to what at which point in a computation's time). We attempt to establish a list of traits an "ideal " MOC should process.
Finally, we describe how the blob model differs from an "ideal " MOC, and identify some natural next steps to achieve such a model.
Keywords : programming, recursion theory, models of computation

Virgile Mogbil - *Parallel Computation with Boolean Proof Nets.*

Abstracts of the Contributions

Clément Aubert - *Proof circuits and others parallel models of computation.*
Proof circuits [Aubert, 11] are a clear and intuitive presentation of the Boolean proof nets [Terui, 04] in a uniform framework [Mogbil-Rahli, 07] : we define /pieces/ as a set of links and edges of a unbounded variant of *Multiplicative Linear Logic* representing Boolean constants, n-ary disjunctions and conjonctions, negation and mechanisms such as deletion and duplication. Thoses pieces may be "plugged" together to obtain /proof circuits/ : *MLLu* uniform Boolean proof nets whose size and depth are implicitly bounded and whose parralel normalization matches up evaluation in Boolean circuits. This light presentation allows sublogarithmic translation and simulation between Boolean circuits and proof circuits, lightens the size of the latter and preserves all the good properties concerning complexity. We conclude by giving the first hints toward a full correspondence between proof circuits and alternating Turing machines, enlarging the correspondence between parallel models of computation.

Ugo Dal Lago - *Higher-order Interpretations and Program Complexity (joint work with Patrick Baillot).*

Polynomial interpretations and their generalizations like quasi-interpretations have been used in the setting of first-order functional languages to design criteria ensuring statically some complexity bounds on programs. This fits in the area of implicit computational complexity, which aims at giving machine-free characterizations of complexity classes. Here we extend this approach to the higher-order setting. For that we consider the notion of simply typed term rewriting systems, we define higher-order polynomial interpretations (HOPI) for them and give a criterion based on HOPIs to ensure that a program can be executed in polynomial time. In order to obtain a criterion which is flexible enough to validate some interesting programs using higher-order primitives, we introduce a notion of polynomial quasi-interpretations, coupled with a simple termination criterion.

Reinhard Kahle - *Computational Complexity and Applicative Theories (joint work with Isabel Oitavem).*
By work of Strahm and Cantini, it was already shown that Applicative Theories - the first order part of Feferman's system of Explicit Mathematics - provide a very handy formal framework to characterize classes of computational complexity. In this talk, we present an applicative theory for FPH and its levels. The presentation will also include some general consideration concerning the set-up of induction principles corresponding to recursion principles for complexity classes.

Martin Lackner - *Fixed-Parameter Algorithms for Finding Minimal Models (joint work with Andreas Pfandler).*
Computing minimal models is an important task in AI and Reasoning that appears in formalisms such as circumscription, diagnosis and answer set programming. Deciding whether there is a minimal model containing a given variable is known to be Σ_2^P-complete.

In this talk I present a study of this problem from the viewpoint of parameterized complexity theory that has been undertaken together with Andreas Pfandler. We performed an extensive complexity analysis of this problem with respect to eleven parameters. We identified tractable fragments based on combinations of these parameters by giving several fixed-parameter algorithms. Furthermore, for the remaining combinations we showed parameterized hardness results and thus proved that no further fixed-parameter algorithms exist for these parameters (under usual complexity theoretic assumptions). In particular, we proved W[2]-completeness when parameterizing by the maximum cardinality of the model. This answered an open question posed in (Gottlob, Scarcello, and Sideri 2002).

Isabel Oitavem - *NP with tier 0 pointers.*
We give a characterization of NP using a recursion scheme with tier 0 point- ers. This extends the Bellantoni-Cook characterization of Ptime and, si- multaneously,

it is a restriction of a recursion-theoretic characterization of Pspace.

Matthieu Perrinel - *Bornes fortes pour la logique linéaire par niveau.*
La logique linéaire par niveaux, introduite par Baillot et Mazza, est un sursystème de la logique linéaire light de Girard. Des bornes polynomiales faibles, c'est à dire pour des stratégies de réduction particulières, ont été montrées pour deux versions de la logique linéaire par niveau ($mL4$ et $mL4_0$). Mais, les stratégies correspondant étant complexe, il était difficile de transformer ces logiques en vrais langages de programmation, avec une stratégie de réduction explicite. En étendant la sémantique des contextes à la logique linéaire par niveaux, nous montrons une borne forte polynomiale (valable pour n'importe quelle stratégie de réduction) pour $mL4$ et $mL4_0$.

Johannes Schmidt - *On the Parameterized Complexity of Default Logic and Autoepistemic Logic (joint work with A. Meier, M. Thomas, and H. Vollmer).*
We investigate the application of Courcelle's Theorem and the logspace version of Elberfeld et al. in the context of the implication problem for propositional sets of formulae, the extension existence problem for default logic, as well as the expansion existence problem for autoepistemic logic and obtain fixed-parameter time and space efficient algorithms for these problems. On the other hand, we exhibit, for each of the above problems, families of instances of a very simple structure that, for a wide range of different parameterizations, do not have efficient fixed-parameter algorithms (even in the sense of the large class XPnu), unless P=NP.

Marco Solieri - *Deep into optimality Complexity and correctness of shared implementation of bounded logics (joint work with Stefano Guerrini and Thomas Leventis) .*
Sharing graphs are an implementation of both the Lévy-optimal reduction on lambda calculus, and of linear logic proof-nets. Since the proof of inadequacy of Lévy families as a cost model for lambda calculus, studies on the complexity of the shared reduction are far from complete. Indeed, the only comparative result considers the restricted case of bounded logics. Because of their well known complexity, it was possible to provide semantically, via geometry of interaction, a correspondent bounding of their shared implementation.

In a similarly restricted case, where the abstract algorithm is sufficient, we present a complete and stronger comparison between the cost of the cut elimination on proof-nets and the cost of the corresponding shared reduction. Then, for the first time, the expected benefits of sharing and avoiding duplication in the reduction are made explicit. The proof exploits an intermediate graph rewriting system, that permits us to give a precise account of complexity on the former and to establish a simulation of the latter. Such simulation implies the main complexity result of the shared implementation, as well as its correctness.

Our syntactical approach enlightens the connection between the different styles of duplication of the two systems - global in proof-nets, local in sharing graphs. This insight appears a suitable starting point for the two, more general, related complexity problem which are still open : the cost model for the lambda calculus itself, and the cost of its shared implementation.

Pierre Valarcher - *From intentional behavior to algorithmic completeness.*
I propose to present problems of the algorithmic expressiveness of some programming languages (imperative and functional) that compute extensionally the same set of functions.