Schedule For Summer Program 2016

Date	9:45 – 11:00	11:30 – 12:45	14:15 – 15:30
May 30	Kamal – Learning Formal Languages	Pradeesha – Topics in Computational Geometry	Raja – Entropy and Counting
May 31	”	”	”
June 1	”	”	Krithika – Special Graph Classes
June 2	Krithika	”	”
June 3	Kamal	”	Kamal
June 6	Teodor – Logic and structures for system modelling	PC Lecture 1	PC Tutorial 1
June 7	”	PC Lecture 2	PC Tutorial 2
June 8	”	PC Lecture 3	PC Tutorial 3
June 9	”	PC Lecture 4	PC Tutorial 4
June 10	”	PC Lecture 5	PC Lecture 6
June 13	Jam – A Theory of Regular “Anything”	Venkatesh: Exact Exponential time algorithms	Ragukumar – The Equitable Coloring of Graphs
June 14	”	”	”
June 15	”	”	Roohani – Complete Disorder is an impossibility
June 16	”	”	”
June 17	”	”	”
June 20	Vikram – Slimy Algorithms	Diptapriyo – Approximation Algorithms for Graph Problems	Anantha & Anup – The Logic-Automata Connection
June 21	”	”	”
June 22	”	Swaroop – The Probabilistic Method	”
June 23	”	”	”
June 24	”	”	”
June 27 onwards	Student Presentations	Student Presentations	

Abstracts of Lectures

Parameterized Complexity (PC) Lectures and Tutorials

Lecture 1: Introduction to Parameterized Complexity and Kernelization (Roohani)
Lecture 2: Bounded Search Trees (Prafullkumar)
Lecture 3: Iterative Compression (Krithika)
Lecture 4: Color Coding (Diptapriyo)
Lecture 5: Treewidth (Roohani)
Lecture 6: Parameterized Intractability (Pranabendu)

Tutorial 1: A Warm-up tutorial (NP-Completeness and Dynamic Programming on trees) (Aditi, Roohani)
Tutorial 2: Kernelization Tutorial (Aditi, Roohani)
Tutorial 3: Bounded Search Trees Tutorial (Sanjukta, Prafullkumar)
Tutorial 4: Iterative Compression Tutorial (Abhishek, Krithika)

Krithika – Special Graph Classes

This set of lectures will introduce graph classes like chordal graphs, bipartite graphs and perfect graphs. We will study structural properties of these graphs and describe algorithms for some classical problems.

Teodor – Logic and structures for system modelling

While for centuries mathematics have been concerned by modelling a continuous world, computer science makes use mainly of discrete methods for the representation, simulation or qualitative prediction of systems behaviours. Computer science logic is of paramount importance in this field because it allows to prove, using appropriate algorithms, that a system, provided its formal description, will or will not evolve according to some property. Abstracting from, on one hand, a user-friendly description of a system and, on the other hand, a syntactic sugar of the logical language, we understand a system as a logical structure and its property (like e.g. "something wrong will never occur") as a logical sentence. May one decide, for a structure $\mathcal{G}$ and a sentence φ, whether $\mathcal{G} \models \varphi$? This is the central question of these few summer lectures.

More precisely, we study several classes of structures and a few classes of sentences. Given such classes $\mathscr{G}$ and Φ, we consider the following decision problem

Instance: $\mathcal{G} \in \mathscr{G}$ and $\varphi \in \Phi$.
Question: $\mathcal{G} \models \varphi$? (or equivalently, $\varphi \in \mathrm{Th}(\mathcal{G})$?)

We are also interested in operations that build one structure from another. For such an operation $\mathfrak{f}$, we wonder whether the decidability of $\mathrm{Th}(\mathfrak{f}(\mathcal{G}))$ may be reduced to the decidability of $\mathrm{Th}(\mathcal{G})$.

A tentative outline of the five lectures is as follows:

1. Introduction (the location of the theme): system defined by a set of sentences, system defined as a single structure
2. Expressing properties of a structure in 1st order logic, 1st order logic with reachability, 1st order logic with transitive closure, monadic 2nd order logic, 2nd order logic
3. Structure building operations and compatibility: interpretations and logical transductions, products, unions, unfolding, iteration, powerset construction
4. Describing an infinite structure: linear structures, branching structures, and more general infinite structures such as transition graphs of machines/automata, graphs of string rewriting systems, graphs of term rewriting systems, transducer-defined structures, automatic structures, Petri nets, equational structures
5. hyperalgebraic structures: higher order equational structures, unfolding hierarchy
6. Deciding theories for a class of structures

Anantha & Anup – The Logic-Automata Connection

The aim of this tutorial is to introduce the connection between 'expressiveness' of two formalisms for describing formal languages, namely, automata and logic.

The first result in this tradition is credited to Buchi and Elgot, who showed that finite word automata and monadic second order logic(interpreted over finite words) have the same expressive

power. We will study this and many such correspondence between classes of formal languages and various logics. We will also study some techniques for showing how a formal language is 'inexpressive' in a certain logic.

Raja – Entropy and Counting

We explain the notion of the entropy of a discrete random variable, and derive some of its basic properties. We then show through few examples how entropy can be useful as a combinatorial enumeration tool.

Pradeesha – Topics in Computational Geometry

We will discuss some topics in Computational Geometry including Convex Hulls, Voronoi Diagram, Triangulation and Range Searching. It will be assumed that the participants have done an introductory course in Data Structures and Algorithms. No prior knowledge of Geometry is required.

Kamal – Learning formal languages

We look at a tiny and basic part of machine learning theory, namely, learning formal languages. Below is an approximate development. Things will change depending on classroom interaction.

Lecture 1: Language acquisition and learning (Big question: how do birds learn how to fly?)
Lecture 2: Deterministic finite automata (Big question: do teachers help?)
Lecture 3: Probabilistic finite automata (Miller and Chomsky, 1963: We cannot seriously propose that a child learns the values of 10^9 parameters in a childhood lasting 10^8 seconds.)
Lecture 4: Substitutability and context-free grammars (Big idea: how far can we push our algorithms?)
Lecture 5: Applications.

Some references are given here `http://www.imsc.res.in/~kamal/tut/learn.html` Some background reading on basics of probability, finite automata and context-free grammars would be required.

Ragukumar – The Equitable Coloring of Graphs

Let the vertices of a graph G be colored with k colors such that no adjacent vertices receive the same color and the sizes of the color classes differ by at most one. Then G is said to be equitably k-colorable. The equitable chromatic number $\chi(G)$ is the smallest integer k such that G is equitably k-colorable. We pay more attention to work done on the Equitable Coloring Conjecture. We also discuss related graph coloring notions and their problems.

Swaroop – The Probabilistic Method

The focus of these lectures will be on learning how to use probabilistic tools in proving mathematical statements. More specifically, we shall concentrate on "Linearity of Expectation" and its applications in proof techniques.
Reference: Extremal Combinatorics by Stasys Jukna , Second edition, Chapter 18.

Jam – A Theory of Regular "Anything"

In automata theory, we learn about "regular" languages, where a language is a set of finite words over a finite alphabet. What is the notion of regularity essentially speaking ? Can we have such notions for sets of trees ? For sets of graphs or other mathematical structures ? And why finite words, what are regular sets of infinite words (and why not infinite trees) ? Going further, why not infinite alphabets as well ? That's an interesting formal game, and rather interestingly the answers can be formulated using simple algebra, with even some game theory somewhere in the

story. But is it all only mathematical entertainment ? As it turns out, such explorations lead us to very practical domains like consistency of XML schema, circuit synthesis and verification of "cyber-physical" systems, offering uniform algorithms in these areas.

Roohani – Complete Disorder is an impossibility

Ever stargazed the sky and noticed that the sky is filled with constellations in the shape of straight lines, rectangles, pentagons and even larger polygons? Could it be that such geometric patterns arise from unknown forces in the cosmos? In 1928, an English mathematician, philosopher and economist, Frank Plumpton Ramsey proved that such patterns are actually implicit in any large structure, whether it is a group of stars, an array of pebbles or a series of numbers generated by throws of a die. In other words, Ramsey proved that complete disorder is an impossibility.

In this series of lectures, we will explore some results around this theme in different settings, for example when the underlying universe is a graph or a set of points in the plane or a sequence of numbers etc. We may also see a connection between results of this kind and the game tic-tac-toe, if time permits.

Vikram – Slimy Algorithms

We understand the mathematical models explaning the unusual collective capability of a single cells organism, called the Physarum Polycephalum, to solve problems such as the shortest path in graphs and approximating Steiner trees.