Implicit computational complexity (ICC) studies machine-independent approaches to computational complexity, with emphasis on approaches based on mathematical logic. Most work consists in characterizations of complexity classes by logical systems (via the Curry-Howard correspondence).

Unfortunately, all these formal systems have been studied with different, often unrelated methodologies, and few results are known about relative intensional expressive power. For example, it is not clear whether combining different systems into one language would break the correspondence to complexity classes. Moreover, two distinct characterizations of the same complexity class do not necessarily capture the same class of programs even if the underlying set of representable functions is the same. Summing up, the area is too fragmented and a paradigm shift is necessary in order to both maintain the research quality at a high level and enforce applications in the context of concrete programming languages (where keeping the class of captured program as large as possible is crucial). What is needed are some unifying syntactic and semantic frameworks for the quantitative analysis of proofs and programs. Such frameworks should be powerful enough to capture many logical systems and programming languages, but simple enough to be useful in evaluating quantitative properties of systems. The research will build over both the well-established “dynamic” semantics of linear logic (e.g., geometry of interaction [Gir89, Gir06] and game semantics [HO95, AM99, MO00]) and new proposals in the area of context semantics [DL05, DL06].

The research program will consist in or more of the following, depending on the candidate.

1. **Implicit computational complexity: Curry-Howard isomorphism and complexity classes**

   Study parametrical logical systems, in such a way that different complexity classes are determined by different parameter choices. Isolate a suitable
notion of “exponential” (i.e., the linear logic modality) so that calculi capturing the different complexity classes can be obtained without changing the rules governing the exponentials, but changing the structural rules only.

Re-examine the different approaches in the literature (in particular the one by Buss based on bounded arithmetic) from a constructive viewpoint, according to the “proofs as programs” paradigm.

2. Intersection types and execution time of programs Intersection types provide a characterization of strongly (resp. weakly) normalizable lambda-terms. This immediately implies type inference to be an undecidable problem. However, restricting to finite-rank intersection enforces decidability while preserving termination. If intersection does not satisfy idempotency, type derivations give more quantitative information on the dynamics of normalization: their size is an upper bound to the number of steps to normal form [Nee04]. Investigate non-idempotent intersection types as a way to study both the degree of intensional expressive power of existing systems in ICC and the intrinsic limits of ICC as a way to capture classes of algorithms as opposed to classes of functions.

3. Type systems and type inference for resource certification Search for large classes of expressible programs (as opposed to the simple definability of functions). Study the problem of type inference for variants of type systems considered so far, extending them with polymorphic, recursive and intersection types. Aim to isolate the largest classes of lambda terms that can be reduced in efficient manner.

4. Quantum model of computation (languages, types and complexity) Aim to define quantum lambda calculi based on the “quantum data + classical control” paradigm. In particular study the structure of quantum computations in an higher-order setting, and prove completeness results with respect to Quantum Turing Machines and Quantum Circuits Families. Define new quantum lambda calculi, in which the quantum register (i.e. quantum data) is ‘embedded’ in the lambda term (versus the notion of configuration used in the previous approaches, where the register is explicitly represented).

Investigate on Quantum Implicit Computational Complexity extending to the quantum case the type-sytems from the well-established field of (classical) Implicit Computational Complexity.

5. Dynamic semantics of linear logic and its “light” fragments Study frameworks in the style of geometry of interaction and context semantics. A geometry of interaction model by Baillot and Pedicini has been exploited to isolate linear logic proofs that can be reduced in elementary time. Study of this approach could be extended to smaller complexity classes.

6. Implicit complexity in a concurrent setting

The very notion of complexity of a computation is not well understood in
a distributed scenario. Study the notion of feasibility for distributed, ubiquitous agents. Adapt techniques well known in the higher-order sequential case to concurrency. Characterize the class of feasible (polynomial?) computation in the concurrent scenario.

References


