http://www.inria.fr/ http://www.cs.unibo.it/ http://www.lix.polytechnique.fr/ http://www.pps.jussieu.fr/

ETERNAL

Interactive Resource Analysis

Motivations

This project aims at putting together ideas from Implicit Computational Complexity and Interactive Theorem Proving, in order to develop new methodologies for handling quantitative properties related to program resource consumption.

Implicit Computational Complexity. The task of verifying and certifying quantitative properties is undecidable as soon as the considered programming language gets close to a general purpose language. So, fully-automatic techniques in general cannot help in classifying programs in a precise way with respect to the amount of resources used. In particular, this is the case for all the techniques based on the study of structural constraints on the shape of programs, like many of those actually proposed in the field of implicit computational complexity.

Proof Assistants. Interactive theorem provers enable the combination of automatic decision procedures and user-guided proof methods. In our framework, undecidability will be handled through the system's user, who is asked not only to write the code, but also to drive the semi-automatic system in finding a proof for the quantitative properties of interest. In order to reduce the user effort and allow him to focus only on the critical points of the analysis, our framework will integrate implicit computational complexity techniques as automatic decision procedures.

People

Focus Team

(INRIA Sophia Antipolis & Università di Bologna)

Ugo Dal Lago
Marco Gaboardi
Simone Martini
Barbara Petit

Parsifal Team

(INRIA Saclay)

Kaustuv Chaudhuri
Dale Miller
Lutz Straßburger

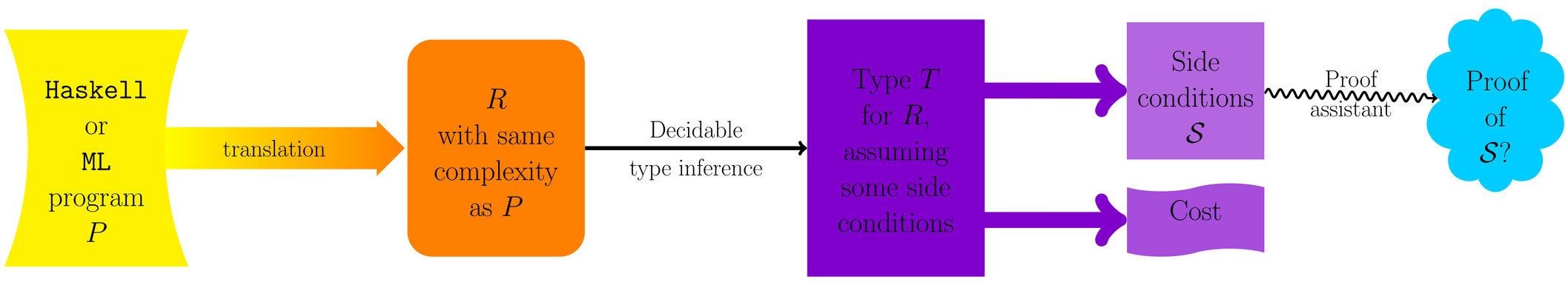
πR^2 Team

(INRIA Rocquencourt & Université Denis Diderot)

Pierre-Louis Curien
Hugo Herbelin
Yann Régis-Gianas
Alexis Saurin

Methodology

We aim to develop an intermediate language (in which "real programs" could be translated), with a type system caring quantitative properties. As such a typing is indecidable, type inference is subject to some *side conditions*, that remain to be proved.



A Typed Language for Resource Consumption. The program is translated to an intermediate language preserving well-typedness and resource consumption. We choose to capture these properties by means of a type system that makes quantitative aspects of programs explicit. The use of an intermediate language allows us to abstract over the original programming language. This intermediate language must be expressive enough, which entails necessarily the non decidability of its typing.

Relative Completeness. The undecidability of type inference for the type system that we will define will be limited to some conditions of combinatorial nature, that we call *side-conditions*. They cannot be verified automatically in general and they are expressed in the language of a rather simple logic, able to express arithmetical statements, like inequalities between polynomials. The previously defined type system, however, must be relatively complete with respect to this logic — a proof of the side conditions would automatically provide a proof of the correctness of the typing.

Interactive and Automatic Resolution of Proof Obligations. One of the main goals of ETERNAL is the study of interactive and automatic techniques for aiding the user in the activity of proving the side conditions written in the aforesaid logic. We expect to be able to derive, from the various existing ICC techniques, some heuristics to assist the user in proving the side-conditions. For example, one of these procedures might be the one induced by the fragment of System F captured by LLL (Light Linear Logic).