# Efficient rewriting
# modulo associativity and commutativity

- Lab: LSV, Cachan, France
- Team: Deducteam
- Advisors: Frédéric Blanqui (INRIA), Gilles Dowek (INRIA and ENS Paris-Saclay) and Gaspard Férey (Mines ParisTech)

**Context.** Lambdapi is a new proof assistant based on a logical framework called the $\lambda\Pi$-calculus modulo rewriting, which is an extension of the simply-typed $\lambda$-calculus (the basis of functional programming languages like OCaml or Haskell) with dependent types (e.g. vectors and matrices of some given dimension) and an equivalence relation on types given by user-defined rewrite rules [1].

This extension is a convenient way to implement computations as terminating confluent rewriting systems. However, in order to encode some algebraic structures, we sometimes have to rely on inherently non-terminating equational theories such as the associativity and commutativity (AC) of some symbols.

**Goal.** The goal of this internship is to implement rewriting with matching modulo AC in Lambdapi and to study potential optimizations to make this new feature usable in practice, for instance in system encodings.

A first implementation could rely on Contejean's certified algorithm [2]. This algorithm is proven correct and complete but it is not deterministic, which means heuristics will have to be tried out and compared. It is rather slow in practice but usable for simple examples.

The non-interactive version of Lambdapi, called Dedukti, rely on the compilation of rewrite rules to decision trees which are known to provide better performances [5]. This feature is not yet implemented in Lambdapi. Studying its impact on Lambdapi's performances with and without rewriting modulo AC could be a first step forward.

Pattern matching modulo AC is known to be NP-complete. Several optimizations are possible in order to improve the practical efficiency. The algorithm could have several heuristics to try out (depth first, breadth first, mixed depending on term shapes) [3, 4]. Study the possibility to remain as lazy as possible to take advantage of sharing and avoid doing the same computations several times.

**Workplan.**

- Study for which kind of rewrite systems the use of decision trees should improve the efficiency of pattern matching.
- Implement pattern matching with decision trees in Lambdapi, setup and run relevant benchmarks to evaluate their actual performance.
- Implement pattern matching modulo AC in Lambdapi without using decision trees, setup and run relevant benchmarks to evaluate the performances. Ideally, this new algorithm should have a negligible impact on Lambdapi's performances if there are no AC symbols.
- Combine both decision trees and pattern matching modulo AC.
- Study further optimizations and heuristics.

**Requirements.** Knowledge of OCaml.

# References


[1] A. Assaf, G. Burel, R. Cauderlier, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard. Dedukti: a logical framework based on the $\lambda\Pi$-calculus modulo theory, 2016. Draft.

[2] E. Contejean. A certified AC matching algorithm. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 3091, 2004.

[3] S. Eker. Fast matching in combinations of regular equational theories. In *Proceedings of the 1st International Workshop on Rewriting Logic and Applications*, Electronic Notes in Theoretical Computer Science 4, 1996.

[4] S. Eker. Associative-Commutative Rewriting on Large Terms. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 2706, 2003.

[5] L. Maranget. Compiling pattern matching to good decision trees. In *Proceedings of the ACM SIGPLAN Workshop on ML*, 2008.