# Post-doctoral researcher position
# Verified data structures for term indexing

February 2, 2023

**Keywords:** automated theorem proving; program verification; term indexing; algorithms; logic

## 1 Research domain

Computation over terms and syntactic expressions is a fundamental operation in computer science, particularly in application domains that rely on mathematical or logical abstractions, such as functional or logic programming languages, automated deduction, rewriting, or symbolic computing. Some programs in these applications deal with a large number of terms, and therefore rely on term indexing data structures and algorithms that are essential to their performance [7]. These data structures are used to quickly retrieve, among a large number of recorded terms, those that satisfy a proprerty, typically unifiability with a given term. These techniques may thus be used to find rewrite rules that can be applied to a term, or to find inference rules that can be applied to a formula. In these applications, the indexes can sometimes contain thousands or even millions or candidates, their optimisation is therefore key to the overall performance of the system. However these algorithms are complex and their implementations may contain faults that are difficult to detect and lead to incorrect behaviors.

## 2 Goal

The aim of this project is to formally verify term indexing algorithms from the literature, using a proof assistant such as Coq [1] or Isabelle [5]. Verifying algorithms in various domains of applications is a standard use of proof assistants, but until now, term indexing data structures and algorithms have not been the object of such work. Although the literature gives some argument as to the correctness of these algorithms, those arguments are typically less detailed than required for a mechanized proof. By providing a formal verification, this project will precisely outline properties of term indexing algorithms that are not always explicitly defined in the literature.

After the verification phase, the code extraction capabilities of the proof assistant [4] will be leveraged to generate a certified software library for term indexing. Special attention will be paid, as early as the formal description of the algorithm, to the performance of extracted code.

# 3 Methodology and existing work

The project will first require identifying target data structures from the literature. The Handbook of Automated Reasoning includes a chapter [7] that surveys numerous term indexing techniques (path indexing, discrimination trees, code trees, substitution trees, etc.) This source is an excellent starting point for this research, however it is not exhaustive and we will have to take into account some techniques that are not mentioned, notably some more recent optimizations (e.g., relational path indexing [6]). We will categorize algorithms according to whether or not they lend themselves to an implemention in a purely functional language, as this characteristic plays an important role for verification. Those algorithms that are suited to purely functional implementation will be the first target for verification, as they lend themselves more naturally to specification in a proof assistant. After verifying these algorithms, we will target fundamentally imperative algorithms and find the best way to represent them in a proof assistant, and to extract efficient code. After verification and code extraction, we will compare the generated code to state-of-the-art non-verified implementations. The aim of this comparison is to test the correctness of these implementations against a verified reference, as well as to measure the performance (index creation time, retrieval time, memory usage) of our code against optimized implementations.

Existing libraries offer formalizations of the mathematical notions that are needed to specify term indexing algorithms (terms, substitutions, unification, etc.) The Archive of Formal Proofs notably offers some Isabelle modules on the meta-theory of logic [2], including a formalization of first-order terms and operations on them, while the CoLoR library [3] for Coq describes the theory of rewriting.

# 4 Candidate profile

This position is open to candidates holding a PhD in computer science or equivalent, or close to the completion date of their PhD. The succesfull candidate will demonstrate scientific skills in one or more of these domains:

- interactive theorem proving
- program verification
- algorithms
- application domains of term indexing techniques (automated deduction, rewriting, functional or logic programming languages, etc.)

# 5 Position

The position is an 18 month contract at Université de Montpellier. The salary before taxes is between 2674€ and 2762€, depending on the candidate's experience. The hiree will work with the MaREL research team, in the LIRMM research department, where they will collaborate with members of the team working on automated deduction and prover implementation. The working language can be either French or English.

To apply, please send a CV, research statement, and up to 3 recommandation letters to Simon Robillard (`simon.robillard@umontpellier.fr`). Prior inquiries about the position are welcome.

# References


[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: the Calculus of Inductive Constructions*. Springer, 2004.

[2] Jasmin Christian Blanchette. Formalizing the metatheory of logical calculi and automatic provers in isabelle/hol (invited talk). In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 1–13, 2019.

[3] Frédéric Blanqui and Adam Koprowski. CoLoR: A Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Mathematical Structures in Computer Science*, 21(04):827–859, 2011.

[4] Pierre Letouzey. Extraction in Coq: an overview. In *Conference on Computability in Europe*, pages 359–369. Springer, 2008.

[5] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283 of *LNCS*. Springer, 2002.

[6] Alexandre Riazanov and Andrei Voronkov. Efficient instance retrieval with standard and relational path indexing. In *Automated Deduction–CADE-19: 19th International Conference on Automated Deduction, Miami Beach, FL, USA, July 28–August 2, 2003. Proceedings 19*, pages 380–396. Springer, 2003.

[7] R. Sekar, I.V. Ramakrishnan, and Andrei Voronkov. Term indexing. In *Handbook of Automated Reasoning*, pages 1853–1964. Elsevier, 2001.