


## Rapport d'évaluation du mémoire de thèse / Evaluation report of the PhD thesis

---

|                    |   |
|--------------------|---|
| <b>Doctorant</b>   | Rodolphe LEPIGRE  |
| <b>PhD student</b> | Mathématiques, sciences et technologie de l'information, informatique (MSTII)<br><i>Semantics and Implementation of an Extension of ML for Proving Programs</i> |
| <b>Rapporteur</b>  | Alexandre Miquel  |
| <b>Reviewer</b>    | University of the Republic (Montevideo, Uruguay)<br>Associate professor ( <i>profesor grado 4</i> )   |

---

### Qualité du mémoire, rédaction & illustrations / Thesis quality, style & illustrations

 Très bon / Very good [X]

*Commentaires/comments :*

The thesis is extremely well written: the presentation of the concepts is clear and pedagogical. All proofs are given and written with great care and the highest level of detail.

---


### Contexte, état de l'art, collaborations / Background, state of the art, collaborations :

*Commentaires/comments : (Good)*

The background is clearly presented, as well as the state of the art.

---

### Qualité scientifique, méthodologie, expérimentations, validation Scientific quality, methodology, experiments, validation

 Très bon / Very good [X]

*Commentaires/comments :*

The thesis is very rich, both scientifically and technically. Moreover, the theoretical work (that is already impressive in itself) is backed by a substantial implementation that proves the applicability of the proposed approach.

---

### Apports personnels, originalité, valorisation, perspectives Personal contributions, originality, valorization, prospects

*Commentaires/comments :*

The proposed approach is very innovative w.r.t. more traditional approaches (e.g. ML languages and type theory), and the presented prototype implementation is really promising.

---

**Conclusions du rapporteur / Reviewer's conclusions**

*Commentaires/comments :*

The material presented in the dissertation fully demonstrates Rodolphe Lepigre's skills to relate the most technical aspects of functional programming with the most advanced theoretical concepts coming from classical realizability. I consider that the dissertation is well worth the title of doctor, and I vividly recommend its defense.

**Avis du rapporteur / Reviewer's opinion :** Favorable ☒ (Très favorable)

Date 18/06/2017

Signature

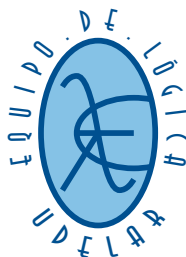


**Visa du directeur de l'école doctorale :**

---

**Rapport détaillé, commentaires libres, questionnements, correction demandées**  
**Detailed report, free comments, questions, requested corrections**

Cf next pages



Alexandre MIQUEL (amiquel@fing.edu.uy)  
Profesor agregado & Maître de conférences (HDR)  
UdelaR / Fing / IMERL / Equipo de Lógica

Montevideo, 18th June 2017

## **Report on Rodolphe LEPIGRE's PhD thesis: *Semantics and Implementation of an Extension of ML for Proving Programs***

The PhD dissertation of Rodolphe LEPIGRE deals with the problem of program verification using computer assisted proofs. In the past four decades, the proofs-as-programs correspondence (a.k.a. the Curry-Howard correspondence) has established strong bridges between functional programming and proof theory. This led to the rise of type theory, a family of expressive type systems where formulas are identified with types and proofs with programs. Many modern proof assistants such as Coq or Agda have been developed on such principles, which make them naturally suited to the verification of purely functional programs.

However, the proof assistants based on type theory traditionally suffer from several limitations —both on the side of programming and on the side of logic— and recently, a lot of research has been done in the direction of proving the correctness properties of broader classes of programs (including non terminating ones), typically by disconnecting (partly or totally) the programming language from the language of proofs.

Following similar motivations, Rodolphe LEPIGRE's PhD dissertation proposes a new and very innovative approach, that consists to start with an untyped functional programming language —namely: an ML-style call-by-value language with variants, records, unlimited recursion and classical control— and then to reconstruct types, formulas and the corresponding notion of proof from a clever analysis of the behaviors that can be observed in a classical realizability model of the untyped language. As a result, the author obtains a very expressive (but undecidable) type system à la Curry, that naturally contains all the ingredients to express formulas and proofs in classical higher-order logic.

## **Synopsis**

Chapter 1 presents the global problematics and the specific problems that will be addressed in the thesis, before concluding with a survey of similar approaches (i.e. the state of the art).

The *untyped* functional programming language at the core of the architecture is presented in Chapter 2, namely: a call-by-value  $\lambda$ -calculus with ML-style records and variants, unrestricted recursion and control operators in the style of Parigot's  $\lambda\mu$ -calculus. The corresponding notion of evaluation is presented using an abstract machine that is directly inspired by Krivine's, with the expected modifications induced by the call-by-value discipline.

Chapter 3 opens with the definition of a first notion of observational equivalence between programs, in the spirit of classical realizability. However, as it will appear in Chapter 5, this basic notion is not sufficient, and the main contribution of the chapter lies in the definition and the study of a broader class of *compatible equivalence relations* (between programs) that can be used as (untyped) notions of observational equivalence.

The type system of the language is introduced in Chapter 4, together with the corresponding realizability semantics. Here, Lepigre takes a radical departure from the standard approach (both in ML languages and in type theory), that traditionally consists to define the type system by a set of inference rules given *a priori*, and then to study its properties (subject reduction, progression, normal forms, etc.) by purely syntactic means. Instead, the type system proposed by Lepigre is completely driven by its semantics, that is given in the form of a classical realizability model (in higher-order logic) following the very spirit of Krivine’s classical realizability, but with a three-layer architecture à la Munch to accommodate the call-by-value strategy.

As it clearly appears in the remaining of the thesis, this radical change of point of view has many advantages. The first advantage is that the type system constructed in this way is correct by construction, in the sense that any program that can be given a particular type necessarily reduces to a value corresponding to that type (at least for pure data types). The second —and perhaps most important— advantage is that it gives a considerable flexibility in the definition of type formers, which include all the expected constructions (arrow types, record types, algebraic types, universal and existential polymorphism, dependent types and equality types), but also provides many new (and sometimes exotic) constructions such as membership (or singleton) types, restriction types, as well as universal and existential quantifications over programs (as opposed to the standard notions of quantification over values). So that in practice, the type system already contains all the ingredients that are needed to represent all the formulas of higher-order logic (as particular types) and all the corresponding proofs in classical logic (as particular programs), which explains why the author does not need to introduce extra constructions to represent them. (In this framework, the proof system comes for free.)

As expected in a higher-order type system à la Curry, the price to pay is the undecidability of typing, although the author claims to have a semi-decision procedure that is sufficient for (type)checking the proofs and programs that are written in practice. (The semi-decision procedure —that relies on the advanced techniques developed in Chapter 6— is not described in the manuscript, but it is already implemented in the accompanying prototype.)

Chapter 5 is devoted to an important problem, known as the problem of *value restriction*. Indeed, it is well known since the very first implementations of SML that *effects* (including control or side effects) interact badly with the rule for introducing polymorphism —a problem that only appears in call-by-value languages, and that is traditionally solved by restricting the application of the *generalization rule* to the programs which happen to be values.

Of course, the same problem arises in Lepigre’s framework, where it is solved by introducing a similar restriction. To prove the correctness of the restricted rule, the author proposes a surprising (and quite elegant) trick that consists to enrich the language of programs with a purely theoretical instruction for testing the observational equivalence between two values, before showing that in the presence of such an instruction, the bi-orthogonal closure operator does not add new values (that is:  $\Phi^{\perp\perp} \cap \Lambda_l^* = \Phi$  for all  $\Phi \subseteq \Lambda_l^*$ ). However, this extension introduces a potential vicious circle between the definition of observational equivalence and the definition of evaluation, which is astutely avoided by distinguishing two notions of observational equivalence that are related using the concepts and results established in Chapter 3.

Chapter 6 introduces many novelties. The first part is devoted to the question of subtyping, that is treated here in a completely renewed way. Indeed, subtyping is not defined as semantic inclusion between sets of values (as one would expect), but as a combination of a new relation of local subtyping with a system of symbolic witnesses corresponding to Hilbert's epsilon construction in logic. Although this new definition is equivalent to the expected definition (which the author forgets to mention), its interest comes from that it allows to reformulate the whole type system without typing contexts, thus facilitating the design of the semi-decision procedure for type checking (that is actually not detailed). Then the author presents the fundamental properties of the type system (completeness on pure data types, normalization, safety and consistency) that immediately follow from the architecture of the realizability model (i.e. correctness by construction). The second part of Chapter 6 is devoted to inductive and coinductive data types (together with the corresponding notions of fixpoints), that are treated in a more standard way, using ordinal sized types (which are reformulated here using the symbolic witnesses introduced in the first part).

In my opinion, Chapter 6 introduces an impressive amount of new material that would have deserved two —if not three— distinct chapters, and it is clear to me that its writing suffered both from a lack of time and from a lack of distance w.r.t. to the treated subject (which is quite inevitable at the end of a PhD). In particular, several important design choices are not fully justified. Nevertheless, this (minor) criticism in the presentation does not undermine neither the interest of the approach nor the soundness of the presented results, whose proofs are still given with an excellent level of detail.

Finally, the concluding Chapter 7 briefly describes the implementation of the programming language PML2 (entirely based on the ideas developed in the thesis), before presenting several examples illustrating the expressiveness of the corresponding type/proof system.

## Overall recommendation

In conclusion, this thesis presents an innovative approach regarding program verification, that is scientifically and technically very rich. The material presented in the dissertation fully demonstrates Rodolphe LEPIGRE's skills to relate the most technical aspects of functional programming with the most advanced theoretical concepts coming from classical realizability. Moreover, the dissertation is extremely well written: the presentation of the concepts is very clear and pedagogical; all proofs are given and written with great care and the highest level of detail. Last but not least, the theoretical work (that is already impressive in itself) is backed by a substantial implementation that proves the applicability of the proposed approach.

For all these reasons, I consider that the dissertation is well worth the title of doctor, and I vividly recommend its defense.

Alexandre Miquel

