Internship proposal: A domain specific language for formally verified legislation

Advisor:
Denis Merigoux
Team Prosecco – Inria
denis.merigoux@inria.fr

Head of department:
Karthikeyan Barghavan
Team Prosecco – Inria
karthikeyan.barghavan@inria.fr

Location of the internship: Paris, France

Context The growing complexity of legislative texts like the French tax code or the more recent European General Data Protection Regulation makes it harder and harder not only to businesses and persons to comply with the law, but also for legislators themselves to reform and craft new legislation compatible with the existing texts. Moreover, some pieces of legislation specify computations and processes that should be defined in a rigorous way: what amount of taxes you should pay according to your revenues, for how long a user's data can be stored and who can access it. Statistical learning techniques can be used to infer some knowledge from the text of the law [8], but they do not provide a systematic and high-assurance way of analysing it. Rather, we prefer to leverage the insights of Kowalski et al. [7] or more recently Lawsky [4, 3], and propose a more formal approach to the problem. The project will build on existing work around the French income tax computation : `https://gitlab.inria.fr/verifisc`. It is also aimed a providing operational solutions which could be used by researchers and/or public institutions.

Goals The goal of this internship would be to develop a Domain Specific Language (DSL) suitable for the specification of computation-heavy legislation in a literate programming style. One source of inspiration for such a DSL could be Business Rules Management Systems such as Drools [6], in order to let users specify the law as it is structured in the text rather than in a more programmatic fashion. However, this DSL would benefit from the modern concepts in programming languages design such as algebraic datatypes, a strong type system and formalized semantics.

The design process, done in collaboration with law professionals, should ensure that the language will easily cope with the evolution of the law. The simplicity of the mathematical fragment used to encode the law should result in a decidable DSL, on which one could use formal methods techniques to tackle problems otherwise intractable in a general setting. Abstract interpretation (for instance using MOPSA [5]), SMT solving (for instance using Z3 [2]), deductive verification (for instance using Coq [1] of F* [9]), etc. could be used to infer meta-properties about the legislation, or to check its coherence.

Qualifications This internship of 6 months or less would be hosted by the team Prosecco at Inria Paris. The preferred qualifications for the student at the beginning of the internship would be:

- fluency with the OCaml programming language;
- experience at writing a compiler for a (preferably functional) language;
- knowledge of how to formalize a programming language's semantics;
- some experience with at least one of the formal methods techniques described above;
- motivation for interacting with potential users of the language.

References

[1] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[2] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[3] Sarah B. Lawsky. Formalizing the Code. *Tax Law Review*, 70(377), 2017.

[4] Sarah B. Lawsky. A Logic for Statutes. *Florida Tax Review*, 2018.

[5] A. Miné, A. Ouadjaout, and M. Journault. Design of a modular platform for static analysis. In *Proc. of 9h Workshop on Tools for Automatic Program Analysis (TAPAS'18)*, Lecture Notes in Computer Science (LNCS), page 4, 28 Aug. 2018. `http://www-apr.lip6.fr/~mine/publi/mine-al-tapas18.pdf`.

[6] Mark Proctor. Drools: A rule engine for complex event processing. In *Proceedings of the 4th International Conference on Applications of Graph Transformations with Industrial Relevance*, AGTIVE'11, pages 2–2, Berlin, Heidelberg, 2012. Springer-Verlag.

[7] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The british nationality act as a logic program. *Commun. ACM*, 29(5):370–386, May 1986.

[8] Harry Surden. Machine learning and law. *Wash. L. Rev.*, 89:87, 2014.

[9] Nikhil Swamy, Catalin Hritcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss, Jean-Karim Zinzindohoué, and Santiago Zanella-Béguelin. Dependent types and multi-monadic effects in F*. In *43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 256–270. ACM, January 2016.