Mlang: an Open-Source Toolchain for the Income Tax Computation[*]

Denis Merigoux[1] and Raphaël Monat[2]

[1] Inria Paris
denis.merigoux@inria.fr
[2] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
raphael.monat@lip6.fr

Abstract

Once per year, the French Directorate for Public Finances (DGFiP) computes for each citizen the amount of income tax owed to the State, depending on their earnings reported on their tax return. This computation is largely automated and performed by a computer program. The source code is written partly in a custom domain specific language called M (used only by the DGFiP), and published yearly. The other part is written in C, and has never been released. Thanks to a mix of retro-engineering and code refactoring, we are able to present Mlang: an open-source toolchain allowing to reproduce the French income tax computation.

Introduction Each year, French citizens declare their earnings to the Directorate of Public Finances (DGFiP). The DGFiP then computes the amount of income tax due from each fiscal household to the State, according to the rules enacted into law by the French Parliament. The computation of the income tax is complex, both in terms of the core calculation (tax brackets and family quotient) and of the astronomical number of exceptions and tax credit that have to be dealt with. The "simplified earnings declaration"[1] has more than 250 inputs. The implementation of the computation has only been partially published[2] since 2016. The published code consists of approximately 92,000 lines of code, and is written in a custom Domain Specific Language (DSL) called "M", created and managed internally by the DGFiP.

Previous work Since the DGFiP did not publish their internal compiler for M, we have retro-engineered the semantics of M[1] with some input from the DGFiP, paving the way for an open-source compiler, Mlang. However, testing the Mlang-compiled published code on private DGFiP test cases yielded a high error rate, with 438 over 542 tests failing. After careful inspection, we concluded we were missing data. In 2020, we were able to sign an agreement with the DGFiP to privately access the source code of the M compiler. It turned out that once the M files have been compiled into C, the compiler inserts those files into a pipeline, which calls the compiled codebase multiple times, and changes the values of some variables between the calls. From a technical point of view, this pipeline is needed because M does not support user-defined functions. This unpublished pipeline, written in C and large of about 33,000 lines of code, accounts for the French tax computation mechanism called *multiples liquidations*. The DGFiP refuses to publish any of its C code, including the pipeline, arguing security concerns.

However, thanks to our private access to the sources, we were able to fix our compiler through the introduction of a new DSL replacing the pipeline, described below. We believe our compiler is now correct for the 2018-income tax computation (except in the case of litigations, happening for 2% of French households[3]).

[*]This work is partially supported by the E.R.C. under Consolidator Grant Agreement 681393 — MOPSA.
[1]https://www3.impots.gouv.fr/simulateur/calcul_impot/2020/simplifie/index.htm
[2]https://framagit.org/dgfip/ir-calcul/
[3]https://www.economie.gouv.fr/files/files/directions_services/dgfip/Rapport/2019/ra2019.pdf

Introducing a new DSL We created a second DSL, called M++, that we used to concisely refactor the unpublished C pipeline into a 100-lines-long new source. Since we considered only the case of tax computations for years after 2018, and where litigations are not supported, the M++ code replaces 6,500 lines of unpublished C code. The downscaling factor for the code size can be explained by writing the code in an appropriate high-level DSL.

Compilation Several transformations (shown in Fig. 1) are applied inside MLANG to the M and M++ code until reaching a common intermediate representation called BIR (Backend Intermediate Representation), which is a simple imperative, function-less language with arithmetic computations and conditional statements. Since our compiler manipulates the whole codebase, it is able to perform optimizations and target multiple backends. Given a restricted set of input and output variables, the generated code is optimized through dead code removal and partial evaluation. These simple optimizations allow us to remove at least 58.5% of the BIR program, and even 97.4% of the program when restricting inputs to the "simplified earnings declaration". We can then translate BIR to Python (our only backend to date) or interpret it.

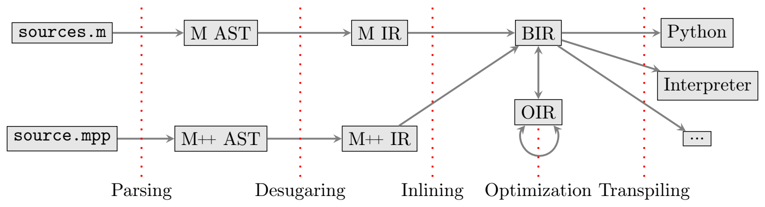

Figure 1: MLANG compilation passes

Correctness of MLANG We evaluated MLANG on the private test set of the DGFiP and passed all of them. We also created our custom test cases using random sampling, and compared the results of MLANG with the internal DGFiP compiler. We did not observe any difference during interpretation of the new test cases.

Conclusion MLANG is an open-source toolchain[4], allowing to reproduce the income tax computation. The whole toolchain is written in OCaml, and is 8,000 LOC, compared to 65,000 LOC for the DGFiP's compiler written in C. We believe that MLANG will be able to replace the ageing compiler used by the DGFiP, which was written in the 1990s. This will allow easier code maintenance, better code analysis support, and the distribution of specialized versions of the income tax computation program in multiple languages. The last point is particularly interesting, as it could increase the level of assurance for a wide range of applications that currently use custom, simplified implementations of the income tax computation like OpenFisca.

References

[1] Denis Merigoux, Raphaël Monat, and Christophe Gaie. Étude formelle de l'implémentation du code des impôts. In *31ème Journées Francophones des Langages Applicatifs*, Gruissan, France, January 2020.

[4]`https://github.com/MLanguage/mlang`