

# Formal Study of the French Tax Code's Implementation

Denis Merigoux & Raphaël Monat

Prosecco, Inria & APR, LIP6, SU

November 4<sup>th</sup>, 2019

**1** Introduction

2 SMT-based legislation study prototype

3 The M language

4 Conclusion

## Code Général des Impôts, Article 197, I, 3, b, 3°

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour [...], le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre :

- au numérateur, la différence entre 20 500 €, pour [...], ou 41 000 €, pour [...], et le montant des revenus mentionnés au troisième alinéa du présent b, et ;
- au dénominateur, 2 000 €, pour [...], ou 4 000 €, pour [...].

## Code Général des Impôts, Article 197, I, 3, b, 3°

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour [...], le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre :

- au numérateur, la différence entre 20 500 €, pour [...], ou 41 000 €, pour [...], et le montant des revenus mentionnés au troisième alinéa du présent b, et ;
- au dénominateur, 2 000 €, pour [...], ou 4 000 €, pour [...].

When does the law specify an algorithm?

## Code Général des Impôts, Article 197, I, 3, b, 3°

Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour [...], le taux de la réduction d'impôt est égal à 20 % multiplié par le rapport entre :

- au numérateur, la différence entre 20 500 €, pour [...], ou 41 000 €, pour [...], et le montant des revenus mentionnés au troisième alinéa du présent b, et ;
- au dénominateur, 2 000 €, pour [...], ou 4 000 €, pour [...].

## When does the law specify an algorithm?

- Decision without human intervention
- No ambiguity
- Quantitative data (income, number of children, etc.)

## An example of algorithmic translation

```
RATE = if (not [...1...]) then  
    20 %
```

“Le taux de la réduction prévue au premier alinéa du présent b est de 20 %. Toutefois, pour [...(1)...],...”

## An example of algorithmic translation

```
RATE = if (not [...1...]) then
  20 %
else
  20 % * (
    (
      )
    /
    (
      )
  )
```

“ il est égal à 20 % multiplié par le rapport entre :

- au numérateur, ...;
- au dénominateur, ...”

## An example of algorithmic translation

```
RATE = if (not [...1...]) then
  20 %
else
  20 % * (
    (
      (if [...2,3...] then 20 500 € else 41000 €)
      - INCOME
    )
    /
    (
      )
  )
)
```

“au numérateur, la différence entre 20 500 €, pour [...(2)...], ou 41 000 €, pour [...(3)...], et le montant des revenus mentionnés au troisième alinéa du présent b”



## An example of algorithmic translation

```
RATE = if (not [...1...]) then
  20 %
else
  20 % * (
    (
      (if [...2,3...] then 20 500 € else 41000 €)
      - INCOME
    )
    /
    (if [...4,5...] then 2000 € else 4000 €)
  )
```

“au dénominateur, 2 000 €, pour [...(4)...], ou 4 000 €, pour [...(5)...].”

# Formal study of the tax computation

What specification for the tax computation?

# Formal study of the tax computation

## What specification for the tax computation?

- inputs : income, number of children, etc
- output =  $f(\text{inputs})$  : amount of tax

$f$  should be studied formally!

# Formal study of the tax computation

## What specification for the tax computation?

- inputs : income, number of children, etc
- output =  $f(\text{inputs})$  : amount of tax

$f$  should be studied formally!

$f$ 's properties:

- Is  $f$  *increasing* with income?
- Is  $f$  *decreasing* with the number of children?

# Formal study of the tax computation

## What specification for the tax computation?

- inputs : income, number of children, etc
- output =  $f(\text{inputs})$  : amount of tax

$f$  should be studied formally!

$f$ 's properties:

- Is  $f$  *increasing* with income?
- Is  $f$  *decreasing* with the number of children?
- What is  $f$ 's *derivative*?  $\Rightarrow$  marginal tax rate

1 Introduction

**2 SMT-based legislation study prototype**

3 The M language

4 Conclusion

## Case study: marginal tax rate

### Model

Characteristic	Household before	Household after
Yearly income	$R_0$	$R_0 + \Delta R$

## Case study: marginal tax rate

### Model

Characteristic	Household before	Household after
Yearly income	$R_0$	$R_0 + \Delta_R$
Income tax	$T_0$	$T_0 + \Delta_T$
Benefits	$B_0$	$B_0 - \Delta_B$



## Case study: marginal tax rate

### Model

Characteristic	Household before	Household after
Yearly income	$R_0$	$R_0 + \Delta_R$
Income tax	$T_0$	$T_0 + \Delta_T$
Benefits	$B_0$	$B_0 - \Delta_B$
Take-home income	$I_0 = R_0 - T_0 + B_0$	$N_0 - \Delta_T - \Delta_B$

## Case study: marginal tax rate

### Model

Characteristic	Household before	Household after
Yearly income	$R_0$	$R_0 + \Delta_R$
Income tax	$T_0$	$T_0 + \Delta_T$
Benefits	$B_0$	$B_0 - \Delta_B$
Take-home income	$I_0 = R_0 - T_0 + B_0$	$N_0 - \Delta_T - \Delta_B$

### Effective withholding marginal rate

$$R_{\text{eff}} = \frac{\Delta_T + \Delta_B}{\Delta_R}$$

## Case study: marginal tax rate

### Model

Characteristic	Household before	Household after
Yearly income	$R_0$	$R_0 + \Delta_R$
Income tax	$T_0$	$T_0 + \Delta_T$
Benefits	$B_0$	$B_0 - \Delta_B$
Take-home income	$I_0 = R_0 - T_0 + B_0$	$N_0 - \Delta_T - \Delta_B$

### Effective withholding marginal rate

$$R_{\text{eff}} = \frac{\Delta_T + \Delta_B}{\Delta_R}$$

Is there a household such that  $R_{\text{eff}} \geq 70\%$ ?

## Our hammer : SMT solving

How to answer that?

# Our hammer : SMT solving

How to answer that?

- Enumerate all possible households?  
⇒ inefficient...

# Our hammer : SMT solving

How to answer that?

- Enumerate all possible households?  
⇒ inefficient...
- Test on real data?  
⇒ fiscal secret, corner cases

# Our hammer : SMT solving

How to answer that?

- Enumerate all possible households?  
⇒ inefficient...
- Test on real data?  
⇒ fiscal secret, corner cases
- Optimization / *machine learning*?  
⇒  $f$  too complex?

# Our hammer : SMT solving

How to answer that?

- Enumerate all possible households?  
⇒ inefficient...
- Test on real data?  
⇒ fiscal secret, corner cases
- Optimization / *machine learning*?  
⇒  $f$  too complex?

## Satisfiability modulo theories

Encoding of the tax computation within bitvector non-linear arithmetic and first-order logic.

⇒ Implemented in a prototype considering a simplified household model



## Counter-example

The SMT solver finds something!

## Counter-example

The SMT solver finds something!

Cohabiting couple with two high-schoolers (15 and 17-years-old), depending on second parent (jobless). Lives in zone II, monthly rent 897,75 €. Monthly raise of 250 €.

## Counter-example

### The SMT solver finds something!

Cohabiting couple with two high-schoolers (15 and 17-years-old), depending on second parent (jobless). Lives in zone II, monthly rent 897,75 €. Monthly raise of 250 €.

Characteristic	Value before	Value after	Variation
Yearly income $R$	33 129,12 €	36 129,12 €	+ 3 000,00 €
IR	3 147,00 €	3 957,00 €	+ 810,00 €
PA	1 320,00 €	0,00 €	- 1 320,00 €
AF	1 584,00 €	1 584,00 €	0,00 €
ARS	806,00 €	0,00 €	- 806,00 €
BC	0,00 €	0,00 €	0,00 €
BL	0,00 €	0,00 €	0,00 €
APL	0,00 €	0,00 €	0,00 €
Take-home income $N$	33 692,12 €	33 756,12 €	+ 64,00 €
Marginal rate			97,9 %

## More questions...

- “Is it possible for a household to win/lose more than  $x$  € after this year’s fiscal reform?”

## More questions...

- “Is it possible for a household to win/lose more than  $x$  € after this year’s fiscal reform?”
- “Is it possible to raise the rate of the second tax bracket by more than 5 points without any household losing more than 200 € yearly?”

## More questions...

- “Is it possible for a household to win/lose more than  $x$  € after this year’s fiscal reform?”
- “Is it possible to raise the rate of the second tax bracket by more than 5 points without any household losing more than 200 € yearly?”
- “Is it possible to find tax bracket rates and fiscal deduction parameters such that the income tax globally brings back 5 billions more yearly, but at the same time no household pays more than 3 % of its income extra?”

## More questions...

- “Is it possible for a household to win/lose more than  $x$  € after this year’s fiscal reform?”
- “Is it possible to raise the rate of the second tax bracket by more than 5 points without any household losing more than 200 € yearly?”
- “Is it possible to find tax bracket rates and fiscal deduction parameters such that the income tax globally brings back 5 billions more yearly, but at the same time no household pays more than 3 % of its income extra?”

<https://gitlab.inria.fr/verifisc/verifisc-python>

# SMT-verified fiscal legislation?

## Avantages

- Systematic analysis
- Very expressive model
- No data needed



# SMT-verified fiscal legislation?

## Avantages

- Systematic analysis
- Very expressive model
- No data needed

## Scalability challenge

- Complexity  $\Rightarrow$  resources and compute time
- SMT queries need optimizing
- Going beyond prototyping

- 1 Introduction
- 2 SMT-based legislation study prototype
- 3 The M language**
- 4 Conclusion

# From prototype to production: leveraging DGFIP's work

The DGFIP<sup>1</sup> released a part of its tax computation system in 2016:

- <https://framagit.org/dgfip/ir-calcul>
- Written in M, custom language
- Computes income taxes for private individuals
- No official, publicly available documentation (no grammar, no semantics, ...)
- 2017 version: 48 files, 92,000 lines of code

---

<sup>1</sup>“Direction Générale des Finances Publiques”: administration in charge of the tax system.

# From prototype to production: leveraging DGFIP's work

The DGFIP<sup>1</sup> released a part of its tax computation system in 2016:

- <https://framagit.org/dgfip/ir-calcul>
- Written in M, custom language
- Computes income taxes for private individuals
- No official, publicly available documentation (no grammar, no semantics, ...)
- 2017 version: 48 files, 92,000 lines of code

## Let's use the DGFIP's work!

- Kept up to date with new legislation by DGFIP
- “Official code”

---

<sup>1</sup>“Direction Générale des Finances Publiques”: administration in charge of the tax system.

# A tax-specific DSL

## Values in M

- Booleans or undef
- Floating-point numbers or undef
- Fixed-size arrays of homogeneous values

# A tax-specific DSL

## Values in M

- Booleans or undef
- Floating-point numbers or undef
- Fixed-size arrays of homogeneous values

## Essence of M

- Statements: mostly variables assignments; exceptions can be raised
- Expressions: thresholds (`min`, `max`) and arithmetic computations
- A *lot* of values  $v \in \{0, 1\}$  used as booleans
- A *few* loops, all having a known, constant iteration number

## A tax-specific DSL: an example

### Variable declaration

```
IRNETBIS : calculee primrest = 0 : "IRNET avant bidouille du 8ZI";
```

# A tax-specific DSL: an example

## Variable declaration

```
IRNETBIS : calculee primrest = 0 : "IRNET avant bidouille du 8ZI";
```

## Computations

```
regle 221220:  
application : iliad ;  
IRNETBIS = max(0, IRNETTER -  
                PIR * positif(SEUIL_12 - IRNETTER + PIR)  
                * positif(SEUIL_12 - PIR)  
                * positif_ou_nul(IRNETTER - SEUIL_12));
```



# A formal semantics for M

- Reverse-engineering of the semantics
- $\mu$ M semantics written in Coq

Looks quite simple:

$$\frac{\text{D-VAR} \quad \Omega(x) = v}{\Omega \vdash x \Downarrow v}$$

$$\frac{\text{D-COND-TRUE} \quad \Omega \vdash e_1 \Downarrow \text{true} \quad \Omega \vdash e_2 \Downarrow v_2}{\Omega \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \Downarrow v_2}$$

$$\frac{\text{D-INDEX} \quad \Omega(x) = (v_0, \dots, v_{n-1}) \quad \Omega \vdash e \Downarrow r \quad r \in \llbracket 0, n-1 \rrbracket}{\Omega \vdash x[e] \Downarrow v_r}$$

# A formal semantics for M

- Reverse-engineering of the semantics
- $\mu$ M semantics written in Coq

Looks quite simple:

$$\frac{\text{D-VAR} \quad \Omega(x) = v}{\Omega \vdash x \Downarrow v}$$

$$\frac{\text{D-COND-TRUE} \quad \Omega \vdash e_1 \Downarrow \text{true} \quad \Omega \vdash e_2 \Downarrow v_2}{\Omega \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \Downarrow v_2}$$

$$\frac{\text{D-INDEX} \quad \Omega(x) = (v_0, \dots, v_{n-1}) \quad \Omega \vdash e \Downarrow r \quad r \in \llbracket 0, n-1 \rrbracket}{\Omega \vdash x[e] \Downarrow v_r}$$

Except for the undef value...

# A formal semantics for M

## The undefined value

Used for:

- default inputs
- runtime errors
- missing cases in inline conditionals

undef-related rules:

$f_1 \odot f_2, \odot \in \{+, -, \times\}$		undef	$f_2 \in \mathbb{F}$
undef		undef	$0 \odot f_2$
$f_1 \in \mathbb{F}$		$f_1 \odot 0$	$f_1 \odot f_2$

# A formal semantics for M

## The undefined value

Used for:

- default inputs
- runtime errors
- missing cases in inline conditionals

undef-related rules:

$f_1 \odot f_2, \odot \in \{+, -, \times\}$	undef	$f_2 \in \mathbb{F}$	$f_1 \div f_2$	undef or 0	$f_2 \in \mathbb{F}, f_2 \neq 0$
undef	undef	$0 \odot f_2$	undef	undef	0
$f_1 \in \mathbb{F}$	$f_1 \odot 0$	$f_1 \odot f_2$	$f_1$	undef	$f_1 \div f_2$

# A formal semantics for M

## The undefined value

Used for:

- default inputs
- runtime errors
- missing cases in inline conditionals

undef-related rules:

$f_1 \odot f_2, \odot \in \{+, -, \times\}$	undef	$f_2 \in \mathbb{F}$	$f_1 \div f_2$	undef or 0	$f_2 \in \mathbb{F}, f_2 \neq 0$
undef	undef	$0 \odot f_2$	undef	undef	0
$f_1 \in \mathbb{F}$	$f_1 \odot 0$	$f_1 \odot f_2$	$f_1$	undef	$f_1 \div f_2$

D-COND-UNDEF

$\Omega \vdash e_1 \Downarrow \text{undef}$

$\Omega \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \Downarrow \text{undef}$

# The Mlang compiler

- Implementation following the semantics
- Input: DGFIP's codebase and query file
- Usecases:
  - Tax computation given input parameters (given in the query file)
  - Compilation to Python, Java; query specifying extraction assumptions
- Written in OCaml
- Freely available: <https://gitlab.inria.fr/verifisc/mlang>

# Code optimization

## Classical compiler optimizations

- Global value numbering
- Dead code elimination
- Partial evaluation

## Simplified simulator case

A case<sup>2</sup>, with less inputs: 2460 to 231 variables

Compiler optimizations: 46,364 variables down to 1,600

---

<sup>2</sup>[https://www3.impots.gouv.fr/simulateur/calcul\\_impot/2019/simplifie/index.htm](https://www3.impots.gouv.fr/simulateur/calcul_impot/2019/simplifie/index.htm)

# Experimental evaluation

The DGFIP provided us with around 500 tests<sup>3</sup>:

- Each test provides inputs and assertions on some variables (median LOC: 597)
- We pass a fifth of the tests (558 to 674 LOC)
- Failed tests: we are missing some constants not provided by the DGFIP
- Awaiting technical answers from the DGFIP

---

<sup>3</sup>They should eventually be public.



## Future work

- Pass all tests
- Translation to Z3 (need to take undefs into account)
- Abstract interpretation for further optimization (Z3 and other)

- 1 Introduction
- 2 SMT-based legislation study prototype
- 3 The M language
- 4 Conclusion**

## Related work

### Academic research

- As old as 1956 [1]!
- Prolog for determining British nationality [7]
- US tax code formalization using default logic [4, 5]
- Smart contracts [3]

# Related work

## Academic research

- As old as 1956 [1]!
- Prolog for determining British nationality [7]
- US tax code formalization using default logic [4, 5]
- Smart contracts [3]

## Private ventures

- Formal vindications [2]
- Imandra [6]
- Legalese

# Conclusion

## Current work and beyond:

- SMT-based tax impact study
- Generating verified code to compute taxes, get it to production
- Upstream formalization to legislative text production

Open-source code: <https://gitlab.inria.fr/verifisc>

JFLA submission: <https://hal.inria.fr/hal-02320347>

Thanks to DGFIP and in particular Christophe Gaie and his team for collaborating!

# References I

- [1] Layman E Allen.  
Symbolic logic: A razor-edged tool for drafting and interpreting legal documents.  
*Yale LJ*, 66:833, 1956.
- [2] D Fernández Duque, M González Bedmar, D Sousa, Joosten, J.J, and G. Errezil Alberdi.  
To drive or not to drive: A formal analysis of requirements (51) and (52) from regulation (eu) 2016/799.  
In *Personalidades jurídicas difusas y artificiales*, pages 159–171. TransJus Working Papers Publication - Edición Especial, 4 2019.
- [3] Tom Hvitved.  
*Contract formalisation and modular implementation of domain-specific languages*.  
PhD thesis, Citeseer, 2011.

## References II

- [4] Sarah B. Lawsky.  
Formalizing the Code.  
*Tax Law Review*, 70(377), 2017.
- [5] Sarah B. Lawsky.  
A Logic for Statutes.  
*Florida Tax Review*, 2018.
- [6] Grant Olney Passmore and Denis Ignatovich.  
Formal verification of financial algorithms.  
In *CADE*, 2017.

## References III

- [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.