Formal Study of the French Tax Code's Implementation

Denis Merigoux & Raphaël Monat & Christophe Gaie

Prosecco, Inria & LIP6, Sorbonne Université & DGFiP

Jan 30th, 2020

Law and algorithms

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 [...].

Law and algorithms

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?

Law and algorithms

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

Get fiscal advice from your local SMT

Who has the highest marginal rate?

Get fiscal advice from your local SMT

Who has the highest marginal rate?

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 €.

Get fiscal advice from your local SMT

Who has the highest marginal rate?

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 %**

SMT-verified fiscal legislation?

Advantages

- Systematic analysis
- Very expressive model
- No data needed

SMT-verified fiscal legislation?

Advantages

- Systematic analysis
- Very expressive model
- No data needed

Scalability challenge

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

From prototype to production: leveraging DGFiP's work

The DGFiP[1] 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

[1] "Direction Générale des Finances Publiques": administration in charge of the tax system.

From prototype to production: leveraging DGFiP's work

The DGFiP[1] 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"

[1]"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
- μM semantics written in Coq

Looks quite simple:

$$\textsc{D-var}\quad \frac{\Omega(x) = v}{\Omega \vdash x \Downarrow v} \qquad \textsc{D-cond-true}\quad \frac{\Omega \vdash e_1 \Downarrow \texttt{true} \qquad \Omega \vdash e_2 \Downarrow v_2}{\Omega \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Downarrow v_2}$$

$$\textsc{D-index}\quad \frac{\Omega(x) = (v_0, \ldots, v_{n-1}) \qquad \Omega \vdash e \Downarrow r \qquad r \in [\![0, n-1]\!]}{\Omega \vdash x[e] \Downarrow v_r}$$

A formal semantics for M

- Reverse-engineering of the semantics
- μM semantics written in Coq

Looks quite simple:

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

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

D-INDEX
$$\frac{\Omega(x) = (v_0, \ldots, v_{n-1}) \qquad \Omega \vdash e \Downarrow r \qquad r \in [\![0, n-1]\!]}{\Omega \vdash x[e] \Downarrow v_r}$$

Except for the `undef` value...

A formal semantics for M

The `undefined` value

Used for:

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

A formal semantics for M

The `undefined` value

Used for:

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

`undef`-related rules:

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

$f_1 \div f_2$	`undef`	$f_2 \in \mathbb{F}, f_2 \neq 0$
`undef`	`undef`	`undef`
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:

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

$f_1 \div f_2$	$\texttt{undef}$	$f_2 \in \mathbb{F}, f_2 \neq 0$
$\texttt{undef}$	$\texttt{undef}$	$\texttt{undef}$
f_1	$\texttt{undef}$	$f_1 \div f_2$

$$\textsc{D-cond-undef}\ \frac{\Omega \vdash e_1 \Downarrow \texttt{undef}}{\Omega \vdash \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \Downarrow \texttt{undef}}$$

The Mlang compiler

- Implementation following the semantics
- Input: DGFiP's codebase and query file
- Usecases:
  - Tax computation given input parameters (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[2], with less inputs: 2460 to 231 variables
Compiler optimizations: 46,364 variables down to 1,600

[2] `https://www3.impots.gouv.fr/simulateur/calcul_impot/2019/simplifie/index.htm`

Experimental evaluation

The DGFiP provided us with around 500 tests[3]:

- 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!
- Awaiting technical answers from the DGFiP...

[3] 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)

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`

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

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

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.

References II

[3] Tom Hvitved.
Contract formalisation and modular implementation of domain-specific languages.
PhD thesis, Citeseer, 2011.

[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.