Formal methods meet legal implementations

Raphaël Monat

rmonat.fr/ecoop_academy26

ECOOP Academy
3 July 2026

Inria

whoami

whoami

Research Scientist

whoami

Research Scientist at Inria Lille

whoami

Research Scientist at Inria Lille/*Rijsel*

whoami

Research Scientist at Inria Lille/*Rijsel* since Sep. 2022.

whoami

Research Scientist at Inria Lille/*Rijsel* since Sep. 2022.

Research Interests

whoami

Research Scientist at Inria Lille/*Rijsel* since Sep. 2022.

Research Interests

- Static analysis: C, Python, multi-language paradigms

whoami

Research Scientist at Inria Lille/*Rijsel* since Sep. 2022.

Research Interests

- Static analysis: C, Python, multi-language paradigms
- Formal methods for public administrations

whoami

Research Scientist at Inria Lille/*Rijsel* since Sep. 2022.

Research Interests

- Static analysis: C, Python, multi-language paradigms
- Formal methods for public administrations
- Open source software, reproducibility & open science

whoami

Research Scientist at Inria Lille/*Rijsel* since Sep. 2022.

Research Interests

- Static analysis: C, Python, multi-language paradigms
- Formal methods for public administrations
- Open source software, reproducibility & open science

Other Research Interests in SyCoMoRES Team

- Scheduling for real-time embedded systems
- Binary code analysis [Bal+19] (for worst-case execution time, security)
- Type systems for privacy, Kleene algebras

Legal implementations?

Software implementations running laws precisely describing computations

Legal implementations?

Software implementations running laws precisely describing computations

- Income tax

Legal implementations?

Software implementations running laws precisely describing computations

- Income tax
- Social benefits

Legal implementations?

Software implementations running laws precisely describing computations

- Income tax
- Social benefits
- Payroll systems

Legal implementations?

Software implementations running laws precisely describing computations

- Income tax
- Social benefits
- Payroll systems
- ~~AI judges~~

Legal implementations are critical software

Critical software with tremendous impact on citizens

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues
  - “$2.2 billion in unplanned expenditures”

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues
  - “$2.2 billion in unplanned expenditures”
- Dutch childcare benefits scandal

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues
  - “$2.2 billion in unplanned expenditures”
- Dutch childcare benefits scandal
  - Wrong accusations of frauds on 10,000s households, reimbursements asked

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues
  - “$2.2 billion in unplanned expenditures”
- Dutch childcare benefits scandal
  - Wrong accusations of frauds on 10,000s households, reimbursements asked
  - “unlawful, discriminatory and improper” – Data Protection Authority [Dut20]

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues
  - “$2.2 billion in unplanned expenditures”
- Dutch childcare benefits scandal
  - Wrong accusations of frauds on 10,000s households, reimbursements asked
  - “unlawful, discriminatory and improper” – Data Protection Authority [Dut20]
  - Culminated with gov. resignation in 2021.

Legal implementations are critical software

Critical software with tremendous impact on citizens

- Phoenix, Canadian payroll system for civil servants [Moc18]
  - Goal: “$70 million in annual savings”
  - Majority of civil servants suffered pay issues
  - “$2.2 billion in unplanned expenditures”
- Dutch childcare benefits scandal
  - Wrong accusations of frauds on 10,000s households, reimbursements asked
  - “unlawful, discriminatory and improper” – Data Protection Authority [Dut20]
  - Culminated with gov. resignation in 2021.

Need more?

See “Automating public services: learning from cancelled systems” [Red+22].

Legal implementations require interdisciplinary collaboration

Law != Computer Science: interdisciplinary collaboration is required.

Legal implementations require interdisciplinary collaboration

Law != Computer Science: interdisciplinary collaboration is required.

- "epistemic trespassing" risk Ballantyne [Bal19] and Lawsky [Law24]
  "Epistemic trespassers are thinkers who have [...] expertise to make good judgments in one field, but move into another field where they lack competence – and pass judgment nevertheless"

Legal implementations require interdisciplinary collaboration

Law != Computer Science: interdisciplinary collaboration is required.

- "epistemic trespassing" risk Ballantyne [Bal19] and Lawsky [Law24]
  "Epistemic trespassers are thinkers who have [...] expertise to make good judgments in one field, but move into another field where they lack competence – and pass judgment nevertheless"
- Experiment where teams implemented a bankruptcy law Escher et al. [Esc+24]

Legal implementations require interdisciplinary collaboration

Law != Computer Science: interdisciplinary collaboration is required.

- "epistemic trespassing" risk Ballantyne [Bal19] and Lawsky [Law24]
  "Epistemic trespassers are thinkers who have [...] expertise to make good judgments in one field, but move into another field where they lack competence – and pass judgment nevertheless"
- Experiment where teams implemented a bankruptcy law Escher et al. [Esc+24]
  - CS or CS+Law teams

Legal implementations require interdisciplinary collaboration

Law != Computer Science: interdisciplinary collaboration is required.

- "epistemic trespassing" risk Ballantyne [Bal19] and Lawsky [Law24]
  "Epistemic trespassers are thinkers who have [...] expertise to make good judgments in one field, but move into another field where they lack competence – and pass judgment nevertheless"
- Experiment where teams implemented a bankruptcy law Escher et al. [Esc+24]
  - CS or CS+Law teams
  - "need for greater skepticism about the [...] reliability of legal software"

Legal implementations require interdisciplinary collaboration

Law != Computer Science: interdisciplinary collaboration is required.

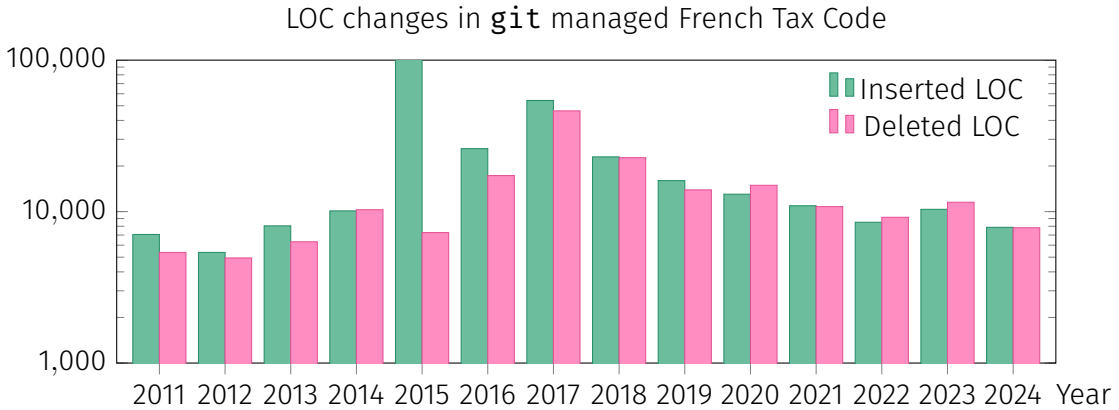
- "epistemic trespassing" risk Ballantyne [Bal19] and Lawsky [Law24]
  "Epistemic trespassers are thinkers who have [...] expertise to make good judgments in one field, but move into another field where they lack competence – and pass judgment nevertheless"
- Experiment where teams implemented a bankruptcy law Escher et al. [Esc+24]
  - CS or CS+Law teams
  - "need for greater skepticism about the [...] reliability of legal software"
  - "participants (esp. CS) expressed unwarranted confidence in [tool utility]"

Legal implementations are updated regularly and incrementally

$\simeq$ yearly legislative changes needing implementations

Legal implementations are updated regularly and incrementally

≃ yearly legislative changes needing implementations



Legal implementations have a long lifespan

⚠ long history not covered here

Legal implementations have a long lifespan

⚠ long history not covered here

French Income Tax

- 1972 First COBOL implementations
- 1987 Transition to codebase still in use

Legal implementations have a long lifespan

⚠ long history not covered here

French Income Tax

- 1972 First COBOL implementations
- 1987 Transition to codebase still in use

French Benefits

- 1996 Centralization of multiple benefits
- Modernization efforts, types of benefits has tripled

Towards high-assurance legal implementations

Trusting the process?

Towards high-assurance legal implementations

Trusting the process?

- Reproducibility of the computation?

Towards high-assurance legal implementations

Trusting the process?

- Reproducibility of the computation?
- Compliance with the law, acting as specification?

Towards high-assurance legal implementations

Trusting the process?

- Reproducibility of the computation?
- Compliance with the law, acting as specification?
- Usability by public administrations, citizens, NGOs, researchers?

Towards high-assurance legal implementations

Trusting the process?

- Reproducibility of the computation?
- Compliance with the law, acting as specification?
- Usability by public administrations, citizens, NGOs, researchers?
- Accurate simulations?

Outline

Joint work

- Marie Alauzen
- Justine Banuls
- Vincent Botbol
- Nicolas Chataing
- Caroline Flori
- Aymeric Fromherz
- Louis Gesbert
- Pierre Goutagny (slides ack)
- Estelle Hary
- Liane Huttner
- Sarah Lawsky
- Denis Merigoux (now transfer lead)
- Romain Primet
- Jonathan Protzenko
- Émile Rolley
- Lilya Slimani

A Modern Compiler for the French Tax Code [MMP21]

Legal implementations

French income tax

- 38M households, 75Md€ of income
- Made public in April 2016 : $\simeq$ 92kLoc M, custom language
- ⚠ Computation not reproducible in 2019

Example M code

Variable declaration

```
IRNETBIS : computed primrest = 0 : "net income tax before 8ZI hack";
8ZI: net tax after living abroad (non-residents)
```

Example M code

Variable declaration

```
IRNETBIS : computed primrest = 0 : "net income tax before 8ZI hack";
8ZI: net tax after living abroad (non-residents)
```

Computation rule

```
rule 221220:
application : iliad ;
IRNETBIS = max(0, IRNETTER -
                  PIR * positive(THRESHOLD_12 - IRNETTER + PIR)
                      * positive(THRESHOLD_12 - PIR)
                      * positive_or_zero(IRNETTER - SEUIL_12));
```

M, briefly

The core of M: **arithmetic expressions** assigned to variables.

M, briefly

The core of M: **arithmetic expressions** assigned to variables.

M quirks

- Static-size arrays (size defined at declaration)

M, briefly

The core of M: **arithmetic expressions** assigned to variables.

M quirks

- Static-size arrays (size defined at declaration)
- Small, unrollable loops

M, briefly

The core of M: **arithmetic expressions** assigned to variables.

M quirks

- Static-size arrays (size defined at declaration)
- Small, unrollable loops
- Use of floating-point numbers, booleans are zero and one

M, briefly

The core of M: **arithmetic expressions** assigned to variables.

M quirks

- Static-size arrays (size defined at declaration)
- Small, unrollable loops
- Use of floating-point numbers, booleans are zero and one
- `undef` value

A formal semantics for M

We reverse-engineered the semantics:

- At first, using a online simulator
- Later, using the private tests DGFiP sent us (Aug. 7, 2019)

A formal semantics for M

We reverse-engineered the semantics:

- At first, using a online simulator
- Later, using the private tests DGFiP sent us (Aug. 7, 2019)

$\Rightarrow$ a μM kernel, its semantics formalized in the Rocq proof assistant.

A formal semantics for M

We reverse-engineered the semantics:

- At first, using a online simulator
- Later, using the private tests DGFiP sent us (Aug. 7, 2019)

$\Rightarrow$ a μM kernel, its semantics formalized in the Rocq proof assistant.

The `undef` value

- Used for: default inputs, runtime errors & missing cases in inline conditionals
- Fun facts: $f + \texttt{undef} = f$, $f \div 0 = 0$, $x[|x| + 1] = \texttt{undef}$, $x[-1] = 0$...

Is this the end?

Test failures

Aug 2019: only 20% of DGFiP tests passed...

Is this the end?

Test failures

Aug 2019: only 20% of DGFiP tests passed...

After investigation, we knew that some code was missing.

Is this the end?

Test failures

Aug 2019: only 20% of DGFiP tests passed...

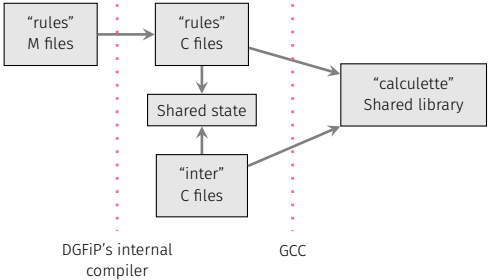
After investigation, we knew that some code was missing.

Patience is key

Aug. 2019	Sent official technical questions to DGFiP
Jan. 2020	Meeting with DGFiP (5 levels of hierarchy involved!)
Apr. 2020	Agreement signed
Jun. 2020	First access to the unpublished sources!

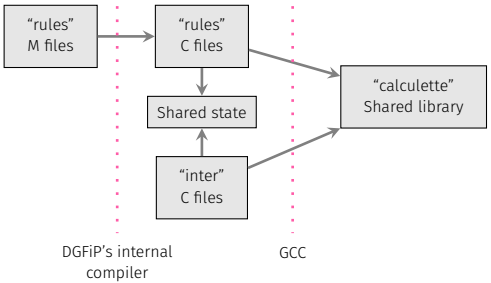
DGFiP's legacy architecture

After 9 months of negotiations, we're in!



DGFiP's legacy architecture

After 9 months of negotiations, we're in!

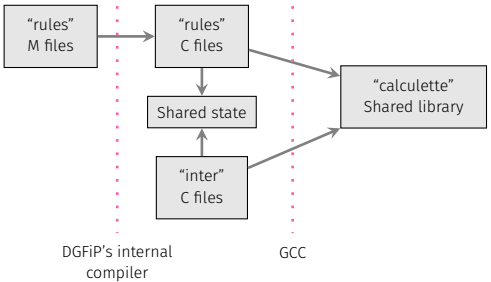


"inter" files

35kLoc of C to bypass M's lack of functions.

DGFiP's legacy architecture

After 9 months of negotiations, we're in!



"inter" files

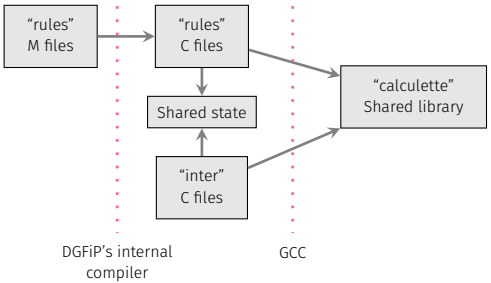
35kLoc of C to bypass M's lack of functions.

Security concerns ⇝ no publication

How to extract the logic of the code?

DGFiP's legacy architecture

After 9 months of negotiations, we're in!



"inter" files

35kLoc of C to bypass M's lack of functions.

Security concerns ⇝ no publication

How to extract the logic of the code?

DSLs to the rescue! Introducing M++

- High-level, no mutable state under the hood
- Tailored for the needs of the "inter" files and DGFiP devs
- 6,000 lines of "inter" C code ⇒ 100 lines of M++

MLANG's architecture

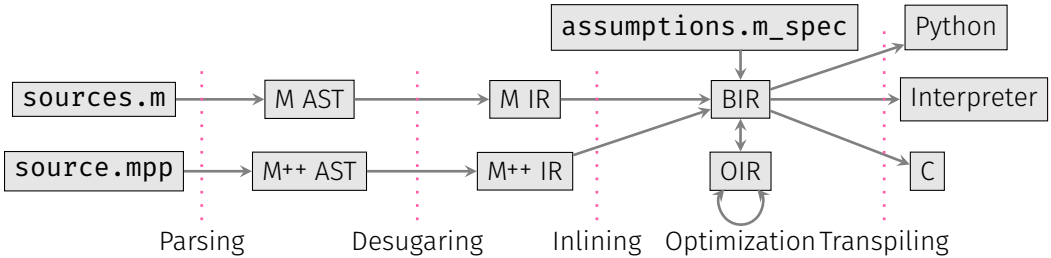
MLANG: written in OCaml, 10k lines of code

`https://github.com/MLanguage/mlang`

MLANG's architecture

MLANG: written in OCaml, 10k lines of code

`https://github.com/MLanguage/mlang`



MLANG's correctness

How to check that MLANG is correct?

- 476 tests from DGFiP
- Generation of our own tests
- Quality measure: value coverage

Mlang's correctness

How to check that Mlang is correct?

- 476 tests from DGFiP
- Generation of our own tests
- Quality measure: value coverage

It works (precise down to the euro)!

All backends validated, on all tests

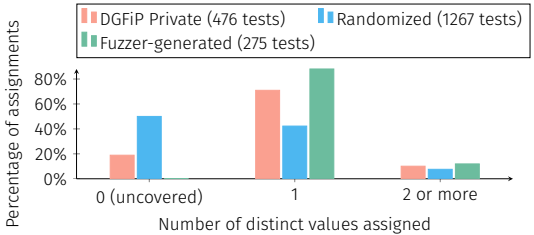
MLang's correctness

How to check that MLang is correct?

- 476 tests from DGFiP
- Generation of our own tests
- Quality measure: value coverage

It works (precise down to the euro)!

All backends validated, on all tests



Code optimization

Compiler optimizations

- Global value numbering
- Dead code elimination
- Partial evaluation
- Dataflow defined-ness analysis

Code optimization

Compiler optimizations

- Global value numbering
- Dead code elimination
- Partial evaluation
- Dataflow defined-ness analysis

Spec. name	# inputs	# outputs	# instructions	% reduction
All	2,459	10,411	129,683	80.2%
Selected outs	2,459	11	99,922	84.8%
Tests	1,635	646	111,839	83.0%
Simplified	228	11	4,172	99.4%
Basic	3	1	553	99.9%

# of instructions with optimizations disabled (2018 code): 656,020.

Code optimization

Compiler optimizations

- Global value numbering
- Dead code elimination
- Partial evaluation
- Dataflow defined-ness analysis

Spec. name	# inputs	# outputs	# instructions	% reduction
All	**2,459**	**10,411**	**129,683**	**80.2%**
Selected outs	2,459	11	99,922	84.8%
Tests	1,635	646	111,839	83.0%
Simplified	228	11	4,172	99.4%
Basic	3	1	553	99.9%

# of instructions with optimizations disabled (2018 code): 656,020.

Contributions around the French tax code

Component	Published by DGFiP?	Contributions
M	yes	Reverse-engineering M $\rightsquigarrow \mu$M in Rocq
“inter” code (C)	no	**DSL** M++ (6kLoc C $\rightsquigarrow$ 100 M++)
M compiler	no	MLANG (10kLoc OCaml) with optimizations
Tests	no	Fuzzing-generated tests (better coverage)

Contributions around the French tax code

Component	Published by DGFiP?	Contributions
M	yes	Reverse-engineering M $\rightsquigarrow$ μM in Rocq
“inter” code (C)	no	**DSL** M++ (6kLoc C $\rightsquigarrow$ 100 M++)
M compiler	no	MLANG (10kLoc OCaml) with optimizations
Tests	no	Fuzzing-generated tests (better coverage)

$\Longrightarrow$ Now reproducible outside DGFiP! [MMP21]

Contributions around the French tax code

Component	Published by DGFiP?	Contributions
M	yes	Reverse-engineering M $\rightsquigarrow \mu$M in Rocq
"inter" code (C)	no	**DSL** M++ (6kLoc C $\rightsquigarrow$ 100 M++)
M compiler	no	MLANG (10kLoc OCaml) with optimizations
Tests	no	Fuzzing-generated tests (better coverage)

$\Longrightarrow$ Now reproducible outside DGFiP! [MMP21]

Interacting with DGFiP

- Long term work: 9 months to access the missing C code
- Pedagogy required: non-technical environment

Industrialization at DGFiP

Research transfer from MLANG to DGFiP, helped by OCamlPro engineers.

Industrialization at DGFiP

Research transfer from MLANG to DGFiP, helped by OCamlPro engineers.

- Getting internal members up to speed on OCaml and MLANG.

Industrialization at DGFiP

Research transfer from MLANG to DGFiP, helped by OCamlPro engineers.

- Getting internal members up to speed on OCaml and MLANG.
- MLANG used in production for most computations since 2023.

Industrialization at DGFiP

Research transfer from MLANG to DGFiP, helped by OCamlPro engineers.

- Getting internal members up to speed on OCaml and MLANG.
- MLANG used in production for most computations since 2023.
- Remediation process in production by 2027.

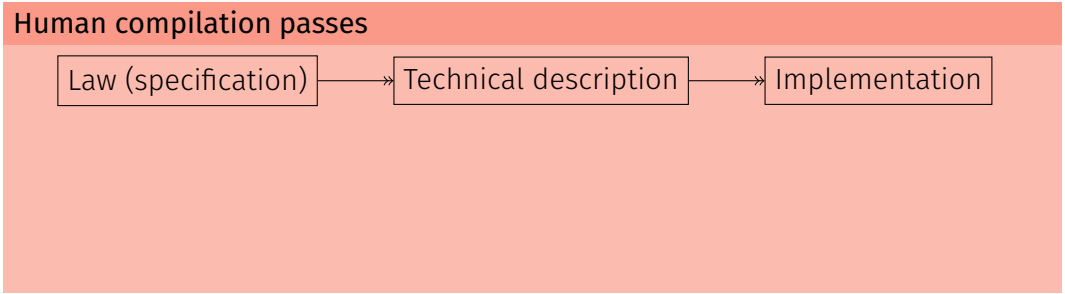
Industrialization at DGFiP

Research transfer from MLANG to DGFiP, helped by OCamlPro engineers.

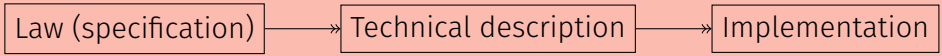
- Getting internal members up to speed on OCaml and MLANG.
- MLANG used in production for most computations since 2023.
- Remediation process in production by 2027.

`https://gitlab.adullact.net/dgfip/impots-nationaux-revenu-patrimoine-particuliers/`

Catala, a DSL for the Law

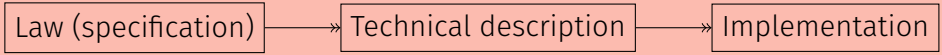

Human compilation passes
Law (specification)
Technical description
Implementation

Human compilation passes



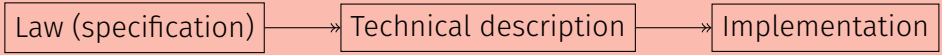
- Structural correspondence is lost in the intermediate representations

Human compilation passes



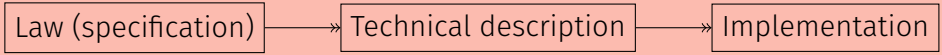
- Structural correspondence is lost in the intermediate representations
- $\rightsquigarrow$ Hidden knowledge, bus factor, difficult to audit, ...

Human compilation passes



- Structural correspondence is lost in the intermediate representations
- $\rightsquigarrow$ Hidden knowledge, bus factor, difficult to audit, ...
- 2019$\rightsquigarrow$2020 : 30% of 90kLoc M changed

Human compilation passes



- Structural correspondence is lost in the intermediate representations
- $\rightsquigarrow$ Hidden knowledge, bus factor, difficult to audit, ...
- 2019$\rightsquigarrow$2020 : 30% of 90kLoc M changed

Catala, a new DSL

Initiated through interdisciplinary work, and requiring peer programming

- Merigoux, Chataing, and Protzenko [MCP21]
- Huttner and Merigoux [HM22]
- Merigoux, Alauzen, and Slimani [MAS23]

Structure of computational law: income tax example

Article 1

The income tax is a fixed percentage of the income.

Structure of computational law: income tax example

Article 1

The income tax is a fixed percentage of the income.

Article 2

The fixed percentage mentioned at article 1 is 20%.

Structure of computational law: income tax example

Article 1

The income tax is a fixed percentage of the income.

Article 2 default case

The fixed percentage mentioned at article 1 is 20%.

Structure of computational law: income tax example

Article 1

The income tax is a fixed percentage of the income.

Article 2

default case

The fixed percentage mentioned at article 1 is 20%.

Article 3

If the income is less than $10,000, the percentage mentioned at article 1 is 10%.

Article 4

For people in charge of 3 or more children, the percentage mentioned at article 1 is 15%.

Structure of computational law: income tax example

Article 1

The income tax is a fixed percentage of the income.

Article 2 default case

The fixed percentage mentioned at article 1 is 20%.

Article 3 exception

If the income is less than $10,000, the percentage mentioned at article 1 is 10%.

Article 4 exception

For people in charge of 3 or more children, the percentage mentioned at article 1 is 15%.

Default logic

Running Example on Income Tax

 # Article 1
The income tax is a fixed percentage of the income.

```catala
scope IncomeTaxComputation:
 definition income_tax equals
   house.income * tax_rate
```

 # Article 2
The fixed percentage mentioned at article 1 is 20%.

```catala
scope IncomeTaxComputation:
  definition tax_rate equals 20%
```

 # Article 3
If the income is less than $10,000, the percentage mentioned at article 1 is 10%.

```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```

 # Article 4
For people in charge of 3 or more children, the percentage mentioned at article 1 is 15%.

```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```

https://book.catala-lang.org/playground/learn.html?chapter=2-1-basic-blocks

Running Example on Income Tax

```
# Article 1
The income tax is a fixed percentage of
the income.
```catala
scope IncomeTaxComputation:
 definition income_tax equals
   house.income * tax_rate
```

Article 2
The fixed percentage mentioned at
article 1 is 20%.
```catala
scope IncomeTaxComputation:
  definition tax_rate equals 20%
```
```

```
# Article 3
If the income is less than $10,000, the
percentage mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```

Article 4
For people in charge of 3 or more
children, the percentage mentioned at
article 1 is 15%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```
```

https://book.catala-lang.org/playground/learn.html?chapter=2-1-basic-blocks

Running Example on Income Tax

```
# Article 1
The income tax is a fixed percentage of
the income.
```catala
scope IncomeTaxComputation:
 definition income_tax equals
   house.income * tax_rate
```

Article 2
The fixed percentage mentioned at
article 1 is 20%.
```catala
scope IncomeTaxComputation:
  definition tax_rate equals 20%
```
```

```
# Article 3
If the income is less than $10,000, the
percentage mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```

Article 4
For people in charge of 3 or more
children, the percentage mentioned at
article 1 is 15%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```
```

https://book.catala-lang.org/playground/learn.html?chapter=2-1-basic-blocks

Compilation into Default Terms

Source code $\xrightarrow{\text{compiler}}$ **default terms**

$$e ::= \langle\, e_1, \ldots, e_n \;\mid\; b_{default} \;:-\; e_{default} \,\rangle$$

Compilation into Default Terms

$$\text{Source code} \xrightarrow{\text{compiler}} \textbf{default terms}$$

$$e ::= \langle \underbrace{e_1, \ldots, e_n}_{\text{exceptions}} \mid b_{default} \ :- \ e_{default} \rangle$$

Compilation into Default Terms

Source code $\xrightarrow{\text{compiler}}$ **default terms**

$$e ::= \langle \underbrace{e_1, \ldots, e_n}_{\text{exceptions}} \mid \underbrace{b_{default}}_{\text{guard}} :- e_{default} \rangle$$

Compilation into Default Terms

Source code $\xrightarrow{\text{compiler}}$ **default terms**

$$e ::= \langle \underbrace{e_1, \ldots, e_n}_{\text{exceptions}} \mid \underbrace{b_{default}}_{\text{guard}} :- \underbrace{e_{default}}_{\text{base case}} \rangle$$

Compilation into Default Terms

Source code $\xrightarrow{\text{compiler}}$ **default terms**

$$e ::= \langle \underbrace{e_1, \ldots, e_n}_{\text{exceptions}} \mid \underbrace{b_{default}}_{\text{guard}} :- \underbrace{e_{default}}_{\text{base case}} \rangle$$

$$\begin{array}{rcl} v & ::= & \texttt{true} \mid \texttt{false} \mid n \mid \ldots \\ & \mid & \varnothing \\ & \mid & \circledast \end{array}$$

Default Terms: Semantics

```
⟨
  ⟨ | income ≤ $10,000 :- 10%⟩,
  ⟨ | nb_children ≥ 3 :- 15%⟩
  | true :- 20%
⟩
```

```
exception definition tax_rate
 under condition house.income <= $10,000
 consequence equals 10%
exception definition tax_rate
 under condition house.nb_children >= 3
 consequence equals 15%
definition tax_rate equals 20%
```

Default Terms: Semantics

$$
\begin{aligned}
&\langle \\
&\quad \langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \\
&\quad \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \\
&\quad \mid \texttt{true :- } 20\% \\
&\rangle
\end{aligned}
$$

```
exception definition tax_rate
 under condition house.income <= $10,000
 consequence equals 10%
exception definition tax_rate
 under condition house.nb_children >= 3
 consequence equals 15%
definition tax_rate equals 20%
```

1. Evaluate all exceptions
2. If **exactly 1 exception** is raised, then **return its value**
3. Else if **at least 2 exceptions** are raised, then **return** $\circledast$
4. Else if **0 exceptions** are raised, evaluate $b_{default}$ and
   - If $b_{default} = \texttt{true}$, then **evaluate** $e_{default}$
   - Else if $b_{default} = \texttt{false}$, then **return** $\varnothing$

Default Terms: Semantics

$$
\begin{array}{l}
\langle \\
\quad \langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } \mathbf{10\%} \rangle, \\
\quad \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \\
\quad \mid \texttt{true} \texttt{ :- } 20\% \\
\rangle
\end{array}
$$

```
exception definition tax_rate
 under condition house.income <= $10,000
 consequence equals 10%
exception definition tax_rate
 under condition house.nb_children >= 3
 consequence equals 15%
definition tax_rate equals 20%
```

1. Evaluate all exceptions
2. If **exactly 1 exception** is raised, then **return its value**
3. Else if **at least 2 exceptions** are raised, then **return** $\circledast$
4. Else if **0 exceptions** are raised, evaluate $b_{default}$ and
   - If $b_{default} = \texttt{true}$, then **evaluate** $e_{default}$
   - Else if $b_{default} = \texttt{false}$, then **return** $\varnothing$

Default Terms: Semantics

$$\langle \quad \langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \quad \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } \mathbf{15\%} \rangle \quad \mid \texttt{true :- } 20\% \quad \rangle$$

```
exception definition tax_rate
 under condition house.income <= $10,000
 consequence equals 10%
exception definition tax_rate
 under condition house.nb_children >= 3
 consequence equals 15%
definition tax_rate equals 20%
```

1. Evaluate all exceptions
2. If **exactly 1 exception** is raised, then **return its value**
3. Else if **at least 2 exceptions** are raised, then **return** $\circledast$
4. Else if **0 exceptions** are raised, evaluate $b_{default}$ and
   - If $b_{default} = \texttt{true}$, then **evaluate** $e_{default}$
   - Else if $b_{default} = \texttt{false}$, then **return** $\varnothing$

Default Terms: Semantics

⟨
⟨ | income ≤ $10,000 :- 10%⟩,
⟨ | nb_children ≥ 3 :- 15%⟩
| true :- 20%
⟩

```
exception definition tax_rate
 under condition house.income <= $10,000
 consequence equals 10%
exception definition tax_rate
 under condition house.nb_children >= 3
 consequence equals 15%
definition tax_rate equals 20%
```

1. Evaluate all exceptions
2. If **exactly 1 exception** is raised, then **return its value**
3. Else if **at least 2 exceptions** are raised, then **return** ⊛
4. Else if **0 exceptions** are raised, evaluate $b_{default}$ and
   - If $b_{default} =$ `true`, then **evaluate** $e_{default}$
   - Else if $b_{default} =$ `false`, then **return** ∅

Default Terms: Semantics

⟨
⟨ | income ≤ $10,000 :- 10%⟩,
⟨ | nb_children ≥ 3 :- 15%⟩
| true :- 20%
⟩

```
exception definition tax_rate
 under condition house.income <= $10,000
 consequence equals 10%
exception definition tax_rate
 under condition house.nb_children >= 3
 consequence equals 15%
definition tax_rate equals 20%
```

1. Evaluate all exceptions
2. If **exactly 1 exception** is raised, then **return its value**
3. Else if **at least 2 exceptions** are raised, then **return** ⊛
4. Else if **0 exceptions** are raised, evaluate $b_{default}$ and
   - If $b_{default} =$ `true`, then **evaluate** $e_{default}$
   - Else if $b_{default} =$ `false`, then **return** ∅

Catala Examples

Multiple examples and prototypes developed by peer-programming processes

Name	Country	Combined LOC
Minimum wage	FR	490
Tax Code (S. 121, 132)	US	637
Family benefits	FR	2,044
Housing benefits	FR	25,290

`https://github.com/CatalaLang/catala-examples/`

Article D823-20 of the French building regulations

The moving allowance is awarded to individuals or households with at least three children born or to be born and who move into a new home entitled to one of the personal housing allowances during a period between the first day of the calendar month following the third month of pregnancy for a child of rank three or more and the last day of the month preceding that in which the child reaches his or her second birthday.
This allowance is payable if the right to assistance is acquired within six months of the date of moving in.

```catala
scope MovingAllowanceEligibility:
 definition condition_moving_period under condition
  (match form.birthdate_third_child_or_more with pattern
  -- MoreThan3Children of date_of_birth_or_pregnancy:
  (match date_of_birth_or_pregnancy with pattern
    --
    --
    -- DateOfBirth of birthday: current_date < (first_day_of_month of (birthday + 2 y))
   )
  )
consequence fulfilled
```

Article D823-20 of the French building regulations

The moving allowance is awarded to individuals or households with at least three children born or to be born and who move into a new home entitled to one of the personal housing allowances during a period between the first day of the calendar month following the third month of pregnancy for a child of rank three or more and the last day of the month preceding that in which the child reaches his or her second birthday.
This allowance is payable if the right to assistance is acquired within six months of the date of moving in.

```
```catala
scope MovingAllowanceEligibility:
 definition condition_moving_period under condition
  (match form.birthdate_third_child_or_more with pattern
  -- MoreThan3Children of date_of_birth_or_pregnancy:
  (match date_of_birth_or_pregnancy with pattern
    --
    --
    -- DateOfBirth of birthday: current_date < (first_day_of_month of (birthday + 2 y))
   )
  )
consequence fulfilled
```
```

Article D823-20 of the French building regulations

The moving allowance is awarded to individuals or households with at least three children born or to be born and who move into a new home entitled to one of the personal housing allowances during a period between the first day of the calendar month following the third month of pregnancy for a child of rank three or more and the last day of the month preceding that in which the child reaches his or her second birthday.
This allowance is payable if the right to assistance is acquired within six months of the date of moving in.

```catala
scope MovingAllowanceEligibility:
 definition condition_moving_period under condition
  (match form.birthdate_third_child_or_more with pattern
  -- MoreThan3Children of date_of_birth_or_pregnancy:
  (match date_of_birth_or_pregnancy with pattern
    -- BeforeFirstDayOfThirdMonthPregnancy: false
    -- AfterFirstDayOfThirdMonthPregnancy: true
    -- DateOfBirth of birthday: current_date < (first_day_of_month of (birthday + 2 y))
   )
  )
consequence fulfilled
```

Outlook: Public Administration Adoption (Merigoux, Gesbert, Primet, Botbol)

Public administration trials

- DGFIP Income Tax Experiment, Jul. 2023-Oct. 2024, 8.9kLOC
- CNAF Family Benefits, Oct. 2024-Oct. 2025, 8.4kLOC

Inria Press Release About Trials

Outlook: Public Administration Adoption (Merigoux, Gesbert, Primet, Botbol)

Public administration trials

- DGFIP Income Tax Experiment, Jul. 2023-Oct. 2024, 8.9kLOC
- CNAF Family Benefits, Oct. 2024-Oct. 2025, 8.4kLOC

Inria Press Release About Trials

Towards adoption at CNAF

CNAF CIO interview

- Emphasis on sovereignty (compared to proprietary solutions), open-source
- Housing benefits, solidarity income, in-work benefit will be migrated

Automated Verification of Catala Programs

Catala is safe

- Designed by PL specialists
- Statically typed
- Decimals for money
- Not Turing complete

Catala is safe

- Designed by PL specialists
- Statically typed
- Decimals for money
- Not Turing complete

But runtime errors can still abound

- New datatypes: dates

Catala is safe

- Designed by PL specialists
- Statically typed
- Decimals for money
- Not Turing complete

But runtime errors can still abound

- New datatypes: dates
- Division by zero, assertion error, etc.

Catala is safe

- Designed by PL specialists
- Statically typed
- Decimals for money
- Not Turing complete

But runtime errors can still abound

- New datatypes: dates
- Division by zero, assertion error, etc.
- **Ambiguities** in the code

Catala is safe

- Designed by PL specialists
- Statically typed
- Decimals for money
- Not Turing complete

But runtime errors can still abound

- New datatypes: dates
- Division by zero, assertion error, etc.
- **Ambiguities** in the code

$\rightsquigarrow$ RTEs unwanted for large batch computations

Ambiguities in Catala code

```
# Article 3
If the income is less than $10,000, the
percentage mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

```
# Article 4
For people in charge of 3 or more children,
the percentage mentioned at article 1 is 15%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```
```

Ambiguities in Catala code

- interpretation conflicts, *e.g.* `income = $9,000` and `children = 4`

Ambiguities in Catala code

```
# Article 3
If the income is less than $10,000, the
percentage mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

```
# Article 4
For people in charge of 3 or more children,
the percentage mentioned at article 1 is 15%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```
```

Ambiguities in Catala code

- interpretation conflicts, *e.g.* `income = $9,000` and `children = 4`
- unhandled cases

Ambiguities in Catala code

```
# Article 3
If the income is less than $10,000, the
percentage mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

```
# Article 4
For people in charge of 3 or more children,
the percentage mentioned at article 1 is 15%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```
```

Ambiguities in Catala code

- interpretation conflicts, *e.g.* `income = $9,000` and `children = 4`
- unhandled cases
- resolved by lawyers/administration if implementation is correct

Fixing the interpretation conflict

Suppose the lawyer says the `income` condition has priority.

```
⟨
  ⟨ | income ≤ $10,000 :- 10%⟩,
  ⟨ | nb_children ≥ 3 :- 15%⟩
  | true :- 20%
⟩
```

Fixing the interpretation conflict

Suppose the lawyer says the `income` condition has priority.
$\rightarrow$ it becomes an exception to the exception.

```
⟨
  ⟨ | income ≤ $10,000 :- 10%⟩,
  ⟨ | nb_children ≥ 3 :- 15%⟩
  | true :- 20%
⟩
```

```
⟨
  ⟨
    ⟨ | income ≤ $10,000 :- 10%⟩
    | nb_children ≥ 3 :- 15%
  ⟩
  | true :- 20%
⟩
```

Automated Verification of Catala Programs

Date Arithmetic [MFM24]

```
$ date -d "2024-01-31 + 1 month" +%F
```

```
$ date -d "2024-01-31 + 1 month" +%F
2024-03-02
```

```
$ date -d "2024-01-31 + 1 month" +%F
2024-03-02
$ date -d "2024-02-01 + 1 month" +%F
```

```
$ date -d "2024-01-31 + 1 month" +%F
2024-03-02
$ date -d "2024-02-01 + 1 month" +%F
2024-03-01
```

```
$ date -d "2024-01-31 + 1 month" +%F
2024-03-02
$ date -d "2024-02-01 + 1 month" +%F
2024-03-01
```

Non-monotonic behavior?!

A wide variety of choices

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)

A wide variety of choices

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
- When do leapers become adults?

A wide variety of choices

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
- When do leapers become adults?
  - 28 February in New Zealand, Taiwan

A wide variety of choices

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
- When do leapers become adults?
  - 28 February in New Zealand, Taiwan
  - 1 March in France, Germany, Hong-Kong

A wide variety of choices

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
- When do leapers become adults?
  - 28 February in New Zealand, Taiwan
  - 1 March in France, Germany, Hong-Kong

$\Longrightarrow$ Formal, flexible semantics required!

A wide variety of choices

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
- When do leapers become adults?
  - 28 February in New Zealand, Taiwan
  - 1 March in France, Germany, Hong-Kong

$\Longrightarrow$ Formal, flexible semantics required! Focus on Gregorian calendar.

Semantics – Values

values	v	$::=$	$(y, m, d) \mid \bot$
date unit	δ	$::=$	$y \mid m \mid d$
expressions	e	$::=$	$v \mid e \ +_\delta \ n$

Semantics – Values

$$
\begin{array}{lrcl}
\text{values} & v & ::= & (y, m, d) \mid \bot \\
\text{date unit} & \delta & ::= & y \mid m \mid d \\
\text{expressions} & e & ::= & v \mid e \ +_\delta \ n
\end{array}
$$

$$
\text{nb\_days}(y, m) = \begin{cases}
29 \text{ if } m = \texttt{Feb} \wedge \text{is\_leap}(y) \\
28 \text{ if } m = \texttt{Feb} \wedge \neg\text{is\_leap}(y) \\
30 \text{ if } m \in \{ \texttt{Apr}, \texttt{Jun}, \texttt{Sep}, \texttt{Nov} \} \\
31 \text{ otherwise}
\end{cases}
$$

Semantics – Values

$$
\begin{array}{lrcl}
\text{values} & v & ::= & (y, m, d) \mid \bot \\
\text{date unit} & \delta & ::= & y \mid m \mid d \\
\text{expressions} & e & ::= & v \mid e \ +_\delta \ n
\end{array}
$$

$$
\text{nb\_days}(y, m) = \begin{cases}
29 \text{ if } m = \texttt{Feb} \wedge \text{is\_leap}(y) \\
28 \text{ if } m = \texttt{Feb} \wedge \neg\text{is\_leap}(y) \\
30 \text{ if } m \in \{ \texttt{Apr}, \texttt{Jun}, \texttt{Sep}, \texttt{Nov} \} \\
31 \text{ otherwise}
\end{cases}
$$

$$
\text{valid}(v) \Leftrightarrow v \neq \bot \wedge v = (y, m, d) \wedge 1 \leq d \leq \text{nb\_days}(y, m)
$$

Semantics – invalid day number

Day additions with invalid day number propagate errors

Semantics – invalid day number

Day additions with invalid day number propagate errors

$$\textsc{Add-Days-Err1}\ \frac{day < 1}{(y, m, day) +_d n \to \bot}$$

Semantics – invalid day number

Day additions with invalid day number propagate errors

$$\textsc{Add-Days-Err1}\quad \frac{day < 1}{(y, m, day) +_d n \to \bot} \qquad \textsc{Add-Days-Err2}\quad \frac{day > \text{nb\_days}(y, m)}{(y, m, day) +_d n \to \bot}$$

Semantics – some cases of month addition

ADD-MONTH

$$\frac{1 \leq mo + n \leq 12}{(y, mo, d) +_m n \to (y, mo + n, d)}$$

Semantics – some cases of month addition

ADD-MONTH

$$\frac{1 \leq mo + n \leq 12}{(y, mo, d) +_m n \to (y, mo + n, d)}$$

ADD-MONTH-OVER

$$\frac{mo + n > 12}{(y, mo, d) +_m n \to (y + 1, mo, d) +_m (n - 12)}$$

Semantics – some cases of month addition

ADD-MONTH

$$\frac{1 \leq mo + n \leq 12}{(y, mo, d) +_m n \to (y, mo + n, d)}$$

ADD-MONTH-OVER

$$\frac{mo + n > 12}{(y, mo, d) +_m n \to (y + 1, mo, d) +_m (n - 12)}$$

Similar cases for ADD-MONTH-UNDER, year, day addition.

Semantics – Rounding

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Semantics – Rounding

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Rounding to valid dates required!

Semantics – Rounding

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Rounding to valid dates required!

$$\begin{array}{lrcl} \text{rounding mode} & r & ::= & \uparrow \mid \downarrow \mid \bot \\ \text{expressions} & e & ::= & v \mid e +_\delta n \mid \mathrm{rnd}_r\, e \end{array}$$

Semantics – Rounding

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Rounding to valid dates required!

$$\begin{array}{lrcl} \text{rounding mode} & r & ::= & \uparrow \mid \downarrow \mid \bot \\ \text{expressions} & e & ::= & v \mid e +_\delta n \mid \mathrm{rnd}_r\, e \end{array}$$

$$\mathrm{rnd}_\uparrow(2024, 02, 31) = (2024, 03, 01)$$

Semantics – Rounding

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Rounding to valid dates required!

$$\begin{array}{lrcl} \text{rounding mode} & r & ::= & \uparrow \mid \downarrow \mid \bot \\ \text{expressions} & e & ::= & v \mid e +_\delta n \mid \mathrm{rnd}_r\, e \end{array}$$

$$\mathrm{rnd}_\uparrow(2024, 02, 31) = (2024, 03, 01)$$
$$\mathrm{rnd}_\downarrow(2024, 02, 31) = (2024, 02, 29)$$

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Rounding to valid dates required!

$$\begin{array}{lrcl} \text{rounding mode} & r & ::= & \uparrow \mid \downarrow \mid \bot \\ \text{expressions} & e & ::= & v \mid e +_\delta n \mid \mathsf{rnd}_r\, e \end{array}$$

$$\mathsf{rnd}_\uparrow(2024, 02, 31) = (2024, 03, 01)$$
$$\mathsf{rnd}_\downarrow(2024, 02, 31) = (2024, 02, 29)$$
$$\mathsf{rnd}_\bot(2024, 02, 31) = \bot$$

$$(2024, 01, 31) +_m 1 \to (2024, 02, 31)$$

Rounding to valid dates required!

$$\begin{array}{lrcl} \text{rounding mode} & r & ::= & \uparrow \mid \downarrow \mid \bot \\ \text{expressions} & e & ::= & v \mid e +_\delta n \mid \mathsf{rnd}_r\, e \end{array}$$

$$\mathsf{rnd}_\uparrow(2024, 02, 31) = (2024, 03, 01)$$
$$\mathsf{rnd}_\downarrow(2024, 02, 31) = (2024, 02, 29)$$
$$\mathsf{rnd}_\bot(2024, 02, 31) = \bot$$

Coreutils-like rounding not defined here

ROUND-NOOP

$$\frac{1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \to (y, m, d)}$$

ROUND-NOOP

$$\frac{1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \to (y, m, d)}$$

ROUND-DOWN

$$\frac{d > \text{nb\_days}(y, m)}{\text{rnd}_\downarrow(y, m, d) \to (y, m, \text{nb\_days}(y, m))}$$

Semantics – Rounding

ROUND-NOOP

$$\frac{1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \to (y, m, d)}$$

ROUND-DOWN

$$\frac{d > \text{nb\_days}(y, m)}{\text{rnd}_\downarrow(y, m, d) \to (y, m, \text{nb\_days}(y, m))}$$

ROUND-UP

$$\frac{d > \text{nb\_days}(y, m) \qquad (y, m, d) +_m 1 \xrightarrow{*} (y', m', d')}{\text{rnd}_\uparrow(y, m, d) \to (y', m', 1)}$$

ROUND-NOOP

$$\frac{1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \to (y, m, d)}$$

ROUND-DOWN

$$\frac{d > \text{nb\_days}(y, m)}{\text{rnd}_\downarrow(y, m, d) \to (y, m, \text{nb\_days}(y, m))}$$

ROUND-UP

$$\frac{d > \text{nb\_days}(y, m) \qquad (y, m, d) +_m 1 \xrightarrow{*} (y', m', d')}{\text{rnd}_\uparrow(y, m, d) \to (y', m', 1)}$$

ROUND-ERR2

$$\frac{d > \text{nb\_days}(y, m)}{\text{rnd}_\bot(y, m, d) \to \bot}$$

Date-period addition

Given a period (ys, ms, ds):

$$e +_r (ys, ms, ds) ::= \mathrm{rnd}_r((e +_y ys) +_m ms) +_d ds$$

Date-period addition

Given a period (ys, ms, ds):

$$e +_r (ys, ms, ds) ::= \mathrm{rnd}_r((e +_y ys) +_m ms) +_d ds$$

Avoids double rounding

Date-period addition

Given a period (ys, ms, ds):

$$e +_r (ys, ms, ds) ::= \mathrm{rnd}_r((e +_y ys) +_m ms) +_d ds$$

Avoids double rounding

Ambiguous expression

A date expression e is ambiguous iff $\mathrm{rnd}_\perp(e) \xrightarrow{*} \perp$

Date-period addition

Given a period (ys, ms, ds):

$$e +_r (ys, ms, ds) ::= \mathrm{rnd}_r((e +_y ys) +_m ms) +_d ds$$

Avoids double rounding

Ambiguous expression

A date expression e is ambiguous iff $\mathrm{rnd}_\perp(e) \xrightarrow{*} \perp$

iff roundings e yield different values

Non-properties

Commutativity of addition

$$(2024, 03, 31) +_{\uparrow} 1\mathrm{m} +_{\uparrow} 1\mathrm{d} = (2024, 05, 01) +_{\uparrow} 1\mathrm{d} = (2024, 05, 02)$$

Non-properties

Commutativity of addition

$$(2024, 03, 31) +_{\uparrow} 1\text{m} +_{\uparrow} 1\text{d} = (2024, 05, 01) +_{\uparrow} 1\text{d} = (2024, 05, 02)$$
$$(2024, 03, 31) +_{\uparrow} 1\text{d} +_{\uparrow} 1\text{m} = (2024, 04, 01) +_{\uparrow} 1\text{m} = (2024, 05, 01)$$

Non-properties

Commutativity of addition

$$(2024, 03, 31) +_{\uparrow} 1\text{m} +_{\uparrow} 1\text{d} = (2024, 05, 01) +_{\uparrow} 1\text{d} = (2024, 05, 02)$$
$$(2024, 03, 31) +_{\uparrow} 1\text{d} +_{\uparrow} 1\text{m} = (2024, 04, 01) +_{\uparrow} 1\text{m} = (2024, 05, 01)$$

“Associativity” of addition

$$(2024, 03, 31) +_{\uparrow} 1\text{m} +_{\uparrow} 1\text{m} = (2024, 05, 01) +_{\uparrow} 1\text{m} = (2024, 06, 01)$$

Non-properties

Commutativity of addition

$$(2024, 03, 31) +_{\uparrow} 1\text{m} +_{\uparrow} 1\text{d} = (2024, 05, 01) +_{\uparrow} 1\text{d} = (2024, 05, 02)$$
$$(2024, 03, 31) +_{\uparrow} 1\text{d} +_{\uparrow} 1\text{m} = (2024, 04, 01) +_{\uparrow} 1\text{m} = (2024, 05, 01)$$

“Associativity” of addition

$$(2024, 03, 31) +_{\uparrow} 1\text{m} +_{\uparrow} 1\text{m} = (2024, 05, 01) +_{\uparrow} 1\text{m} = (2024, 06, 01)$$
$$(2024, 03, 31) +_{r} 2\text{m} = (2024, 05, 31)$$

Formalized properties

All formalized with the F* proof assistant.

- More in the paper [MFM24].

Formalized properties

All formalized with the F* proof assistant.

- More in the paper [MFM24].
- During our study, we used QCheck to test our intuition.

Formalized properties

All formalized with the F⋆ proof assistant.

- More in the paper [MFM24].
- During our study, we used QCheck to test our intuition.

Well-formedness

For any date d, any period p, any value v, and $r \in \{\downarrow, \uparrow\}$, we have:

$$\text{valid}(d) \wedge d +_r p \xrightarrow{*} v \Rightarrow \text{valid}(v)$$

Formalized properties

All formalized with the F* proof assistant.

- More in the paper [MFM24].
- During our study, we used QCheck to test our intuition.

Well-formedness

For any date d, any period p, any value v, and $r \in \{\downarrow, \uparrow\}$, we have:

$$\text{valid}(d) \wedge d +_r p \xrightarrow{*} v \Rightarrow \text{valid}(v)$$

Date addition is monotonic

For any dates d_1, d_2, period p, $r \in \{\downarrow, \uparrow\}$, if $d_1 < d_2$, then $d_1 +_r p \leq d_2 +_r p$

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe
- Otherwise, the rounding of `d + 1 month` will not change the comparison.

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe
- Otherwise, the rounding of `d + 1 month` will not change the comparison.

$\Longrightarrow$ Prove rounding-insensitivity of an expression e

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe
- Otherwise, the rounding of `d + 1 month` will not change the comparison.

$\Longrightarrow$ Prove rounding-insensitivity of an expression e

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe
- Otherwise, the rounding of `d + 1 month` will not change the comparison.

$\Longrightarrow$ Prove rounding-insensitivity of an expression e

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$ encoded as $\mathtt{sync}(e)$

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe
- Otherwise, the rounding of `d + 1 month` will not change the comparison.

$\Longrightarrow$ Prove rounding-insensitivity of an expression e

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$ encoded as $\mathtt{sync}(e)$
- Considering product programs with both rounding modes [DOM21]

Meaningful ambiguities

Rounding choice can change comparisons

```
d + 1 month <= April 30 2024
```

- Rounding-sensitive comparison `d = March 31 2024`

When rounding up or down doesn't change a computation

```
d + 1 month <= April 15 2024
```

- No rounding? Safe
- Otherwise, the rounding of `d + 1 month` will not change the comparison.

$\Longrightarrow$ Prove rounding-insensitivity of an expression e

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$ encoded as $\texttt{sync}(e)$
- Considering product programs with both rounding modes [DOM21]
- Will reduce the need for costly legal interpretations

- Defines addition, accessors, projection, lexicographic comparison

- Defines addition, accessors, projection, lexicographic comparison
- Translates constraints on dates into numerical constraints

- Defines addition, accessors, projection, lexicographic comparison
- Translates constraints on dates into numerical constraints
  date $d_1 \rightsquigarrow$ ghost numerical variables $\mathsf{d}(d_1), \mathsf{m}(d_1), \mathsf{y}(d_1)$

- Defines addition, accessors, projection, lexicographic comparison
- Translates constraints on dates into numerical constraints
  date $d_1 \rightsquigarrow$ ghost numerical variables $\mathsf{d}(d_1), \mathsf{m}(d_1), \mathsf{y}(d_1)$
- Acts as a functor lifting a numerical abstract domain

- Defines addition, accessors, projection, lexicographic comparison
- Translates constraints on dates into numerical constraints
  date $d_1 \rightsquigarrow$ ghost numerical variables $\mathsf{d}(d_1), \mathsf{m}(d_1), \mathsf{y}(d_1)$
- Acts as a functor lifting a numerical abstract domain

$\mathsf{d}(d_1) \in [1, 31] \wedge \mathsf{m}(d_1) \in [1, 12] \wedge \mathsf{y}(d_1) = 2024$: all valid dates of 2024

Transfer function computing $(\mathtt{d}, \mathtt{m}, \mathtt{y}) +^{\#} \mathtt{nb\_m}$ in abstract state `abs`

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
```

Transfer function computing $(d, m, y) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define symbolic expressions for the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
```

Transfer function computing $(\mathtt{d}, \mathtt{m}, \mathtt{y}) +^{\#} \mathtt{nb\_m}$ in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define symbolic expressions for the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d = 31 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;






  ]
```

Transfer function computing $(d, m, y) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define symbolic expressions for the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d = 31 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*) (*^^^^^^^^^^^^^^*)
    (*********** guard condition *************) (* continuation *)




  ]
```

Transfer function computing $(\mathsf{d}, \mathsf{m}, \mathsf{y}) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define symbolic expressions for the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d = 31 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*) (*^^^^^^^^^^^^^^*)
    (*********** guard condition *************) (* continuation *)
    (* ~> continuation (assume guard) *)



  ]
```

Transfer function computing $(\mathtt{d}, \mathtt{m}, \mathtt{y}) +^{\#} \mathtt{nb\_m}$ in abstract state `abs`

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define symbolic expressions for the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d = 31 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (* Case 2: round resulting date to 28/02/Y, Y is not leap *)
    d > 28 && r_m = Feb && not (is_leap r_y), round 28 r_m r_y;
    (* Case 3: round resulting date to 29/02/Y, Y is leap *)
    d > 29 && r_m = Feb && is_leap r_y, round 29 r_m r_y;


  ]
```

Transfer function computing $(\mathtt{d}, \mathtt{m}, \mathtt{y}) +^{\#} \mathtt{nb\_m}$ in abstract state `abs`

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define symbolic expressions for the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d = 31 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (* Case 2: round resulting date to 28/02/Y, Y is not leap *)
    d > 28 && r_m = Feb && not (is_leap r_y), round 28 r_m r_y;
    (* Case 3: round resulting date to 29/02/Y, Y is leap *)
    d > 29 && r_m = Feb && is_leap r_y, round 29 r_m r_y;
    (* Case 4: no rounding *)
    mk_true, mk_date d r_m r_y;
  ]
```

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathtt{d}(d1) = 31,\ \mathtt{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Oct}\}, \mathtt{d}(d2) = 30,\ \mathtt{m}(d2) = \mathtt{m}(d1) + 1, \mathtt{y}(d2) = \mathtt{y}(d1)$$

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathtt{d}(d1) = 31,\ \underbrace{\mathtt{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Oct}\}}_{\text{Bounded set of ints}}, \mathtt{d}(d2) = 30,\ \mathtt{m}(d2) = \mathtt{m}(d1) + 1, \mathtt{y}(d2) = \mathtt{y}(d1)$$

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathtt{d}(d1) = 31,\ \underbrace{\mathtt{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Oct}\}}_{\text{Bounded set of ints}}, \mathtt{d}(d2) = 30,\ \underbrace{\mathtt{m}(d2) = \mathtt{m}(d1) + 1, \mathtt{y}(d2) = \mathtt{y}(d1)}_{\text{Polyhedra}}$$

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathtt{d}(d1) = 31,\ \underbrace{\mathtt{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Oct}\}}_{\text{Bounded set of ints}}, \mathtt{d}(d2) = 30,\ \underbrace{\mathtt{m}(d2) = \mathtt{m}(d1) + 1, \mathtt{y}(d2) = \mathtt{y}(d1)}_{\text{Polyhedra}}$$

- No rounding $\quad \mathtt{d}(d1) = \mathtt{d}(d2),\ \mathtt{m(d2)} \equiv_{12} \mathtt{m(d1)} + 1, \mathtt{y}(d1) \leq \mathtt{y}(d2) \leq \mathtt{y}(d1) + 1$

`date d1 = rand_date(); date d2 = d1 + 1 month;` rounding down.

- No concrete constraints on `d1`
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathtt{d}(d1) = 31,\ \underbrace{\mathtt{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Oct}\}}_{\text{Bounded set of ints}}, \mathtt{d}(d2) = 30,\ \underbrace{\mathtt{m}(d2) = \mathtt{m}(d1) + 1, \mathtt{y}(d2) = \mathtt{y}(d1)}_{\text{Polyhedra}}$$

- No rounding $\mathtt{d}(d1) = \mathtt{d}(d2),\ \underbrace{\mathtt{m(d2)} \equiv_{12} \mathtt{m(d1)} + 1}_{\text{Linear congruence domain}}, \mathtt{y}(d1) \leq \mathtt{y}(d2) \leq \mathtt{y}(d1) + 1$

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

```
date d1 = rand_date(); date d2 = d1 + 1 month;
```
double semantics

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

`date d1 = rand_date(); date d2 = d1 + 1 month;` double semantics

- No rounding

$$\mathsf{d}(d1) = \mathsf{d}(d2) \qquad \mathsf{m}(\mathtt{d2}) \equiv_{12} \mathsf{m}(\mathtt{d1}) + 1 \qquad \mathsf{y}(d1) \leq \mathsf{y}(d2) \leq \mathsf{y}(d1) + 1$$

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

`date d1 = rand_date(); date d2 = d1 + 1 month;` double semantics

- No rounding

$$\mathsf{d}(d1) = \mathsf{d}(d2) \qquad \mathsf{m}(\mathtt{d2}) \equiv_{12} \mathsf{m}(\mathtt{d1}) + 1 \qquad \mathsf{y}(d1) \leq \mathsf{y}(d2) \leq \mathsf{y}(d1) + 1$$

- 30-day month

$$\mathsf{d}(d1) = 31, \mathsf{m}(d1) \in \{\, \mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Sep} \,\}$$
$$\downarrow\mathsf{d}(d2) = 30, \downarrow\mathsf{m}(d2) \in \{\, \mathtt{Apr}, \mathtt{Jun}, \mathtt{Sep}, \mathtt{Nov} \,\}, \downarrow\mathsf{m}(d2) = \mathsf{m}(d1) + 1$$
$$\uparrow\mathsf{d}(d2) = 1, \uparrow\mathsf{m}(d2) \in \{\, \mathtt{May}, \mathtt{Jul}, \mathtt{Oct}, \mathtt{Dec} \,\}, \uparrow\mathsf{m}(d2) = \mathsf{m}(d1) + 2$$
$$\downarrow\mathsf{y}(d2) = \uparrow\mathsf{y}(d2) = \mathsf{y}(d1)$$

Implementation into Mopsa 

- Open-source static analysis platform

Implementation into Mopsa 

- Open-source static analysis platform
- C, Python, C+Python programs

Implementation into Mopsa 

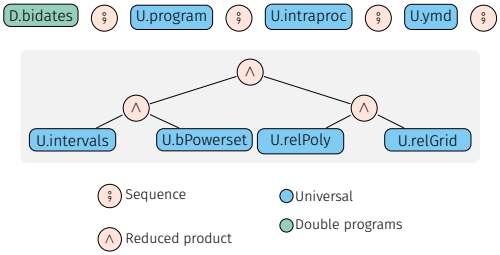
- Open-source static analysis platform
- C, Python, C+Python programs
- `gitlab.com/mopsa/mopsa-analyzer`

Implementation into Mopsa 

- Open-source static analysis platform
- C, Python, C+Python programs
- `gitlab.com/mopsa/mopsa-analyzer`
- Winner of *SoftwareSystems* @ SV-Comp'26

Implementation into Mopsa

- Open-source static analysis platform
- C, Python, C+Python programs
- `gitlab.com/mopsa/mopsa-analyzer`
- Winner of *SoftwareSystems* @ SV-Comp'26



Extracted sample from French housing benefits

```
date current = rand_date();
date birthday = rand_date();
date intermediate = birthday + [2 years, 0 months, 0 days];
date limit = first_day_of(intermediate);
assert(sync(current < limit));
```

Extracted sample from French housing benefits

```
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

```
5: assert(sync(current < limit));
                ^^^^^^^^^^^^^^^^
```

```
Desynchronization detected: (current < limit). Hints:
↑month(limit) = 3, ↑day(limit) = 1, ↓month(limit) = 2, ↓day(limit) = 1,
↑month(intermediate) = 3, ↑day(intermediate) = 1, ↓month(intermediate) = 2,
↓day(intermediate) = 28, month(birthday) = 2, day(birthday) = 29,
year(birthday) =[4] 0, month(current) = 2, day(current) = [1,29],
year(current) = ↑year(intermediate) = ↑year(limit)
= ↓year(intermediate) = ↓year(limit) = year(birthday) + 2
```

Extracted sample from French housing benefits

```
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

```
5: assert(sync(current <
                ^^^^^^^^^
```

Computed, actual counter-example

```
Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) =
```

Extracted sample from French housing benefits

```
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

```
5: assert(sync(current <
                  ^^^^^^^^^^
Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) =
```

Computed, actual counter-example

- `current` is in Feb. of year *y*

Extracted sample from French housing benefits

```
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

```
5: assert(sync(current <
                  ^^^^^^^^^
Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) =
```

Computed, actual counter-example

- `current` is in Feb. of year y
- `birthday` is 29 Feb. of leap year $y - 2$

Extracted sample from French housing benefits

```
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

```
5: assert(sync(current <
                  ^^^^^^^^^^
Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) =
```

Computed, actual counter-example

- `current` is in Feb. of year y
- `birthday` is 29 Feb. of leap year $y - 2$
- `intermediate` is either 28 Feb. or 1 March of y

Extracted sample from French housing benefits

```
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

```
5: assert(sync(current <
                  ^^^^^^^^^^
Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) =
```

Computed, actual counter-example

- `current` is in Feb. of year y
- `birthday` is 29 Feb. of leap year $y - 2$
- `intermediate` is either 28 Feb. or 1 March of y
- `limit` is either 1 Feb. or 1 March of y

Contributions to Catala

- Date-rounding library `dates-calc`

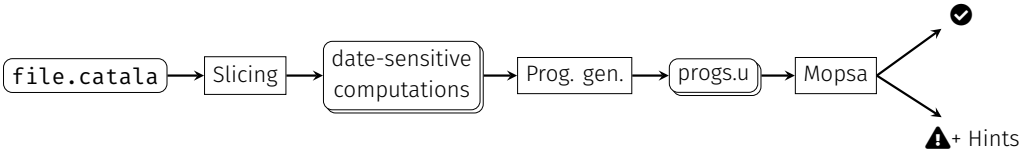
Contributions to Catala

- Date-rounding library `dates-calc`
- Scope-level rounding mode configuration

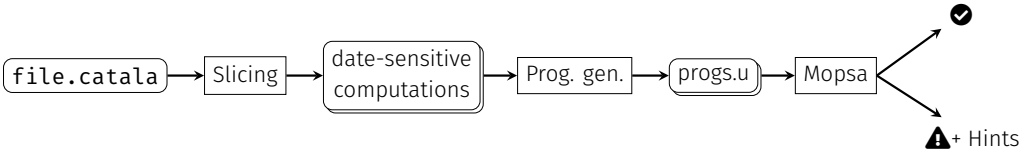
Contributions to Catala

- Date-rounding library `dates-calc`
- Scope-level rounding mode configuration
- Connection with static analysis

Date ambiguity detection pipeline

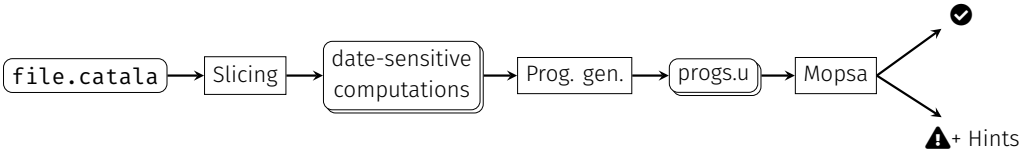


Date ambiguity detection pipeline



2 rounding-sensitive cases detected

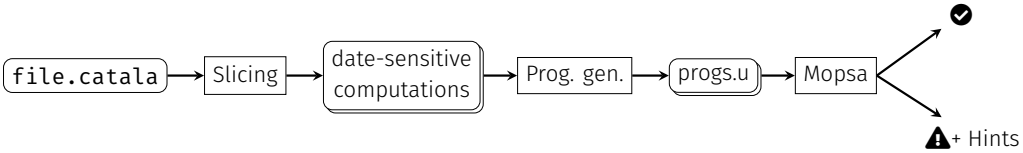
Date ambiguity detection pipeline



2 rounding-sensitive cases detected

No false alarms

Date ambiguity detection pipeline

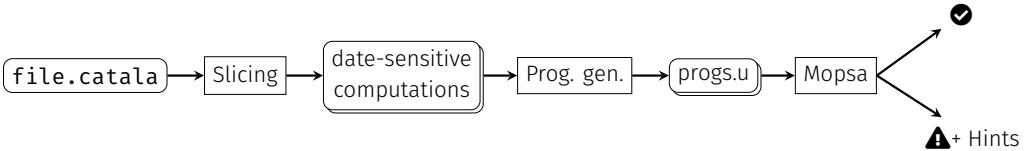


2 rounding-sensitive cases detected

No false alarms

Intra-scope extraction for now

Date ambiguity detection pipeline



2 rounding-sensitive cases detected

No false alarms

Intra-scope extraction for now

Manual inter-scope extraction

16 additional cases:

- 10 can be proved safe assuming `current_date` $\geq$ `2023`
- Other are real issues

What about mainstream implementations?

Survey of implementations

- Java, `boost` round down
- Python `stdlib`: no month addition
- `boost` C++ rounds down
- Rounding inconsistency in spreadsheet operators

Automated Verification of Catala Programs

CuteCat Concolic Execution [GFM25]

Concolic execution: first example

```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```

Concolic execution: first example

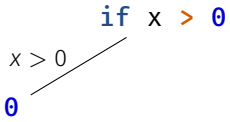
```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```

```
if x > 0
```

Step	x	y	Output	Constraints after evaluation	Next path to try

Concolic execution: first example

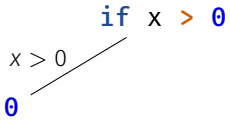
```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```



Step	x	y	Output	Constraints after evaluation	Next path to try
1	1	20	0	$x > 0$	

Concolic execution: first example

```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```

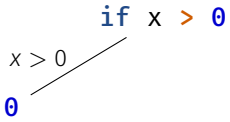


Step	x	y	Output	Constraints after evaluation	Next path to try
1	1	20	0	$x > 0$	$\neg(x > 0)$

↻ SMT

Concolic execution: first example

```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```

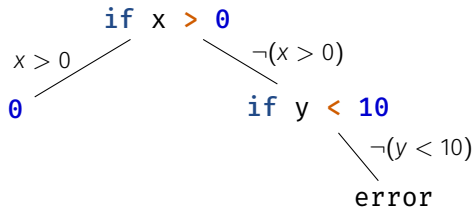


Step	x	y	Output	Constraints after evaluation	Next path to try
1	1	20	0	$x > 0$	$\neg(x > 0)$
2	0	20			

↻ SMT

Concolic execution: first example

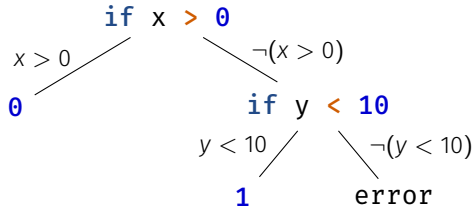
```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```



Step	x	y	Output	Constraints after evaluation	Next path to try	
1	1	20	0	$x > 0$	$\neg(x > 0)$	↻ SMT
2	0	20	`error`	$\neg(x > 0) \wedge \neg(y < 10)$	$\neg(x > 0) \wedge y < 10$	↻ SMT

Concolic execution: first example

```
if x > 0
    then 0
    else if y < 0
        then 1
        else error
```



Step	x	y	Output	Constraints after evaluation	Next path to try	
1	1	20	0	$x > 0$	$\neg(x > 0)$	↻ SMT
2	0	20	error	$\neg(x > 0) \wedge \neg(y < 10)$	$\neg(x > 0) \wedge y < 10$	↻ SMT
3	0	9	1	$\neg(x > 0) \wedge y < 10$	-	

Concolic execution of default terms

$$\langle\langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \mid \texttt{true} \texttt{ :- } 20\% \rangle$$

Concolic execution of default terms

$\langle\langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \mid \texttt{true} \texttt{ :- } 20\% \rangle$

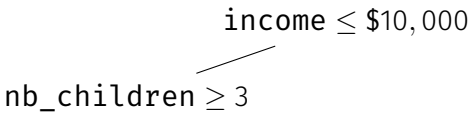
$\texttt{income} = \$9{,}000; \quad \texttt{nb\_children} = 4$

$\texttt{income} \leq \$10,000$

Concolic execution of default terms

$\langle\langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \mid \texttt{true} \texttt{ :- } 20\% \rangle$

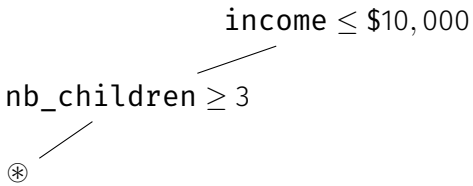
$\texttt{income} = \$9{,}000; \quad \texttt{nb\_children} = 4$



Concolic execution of default terms

⟨⟨ | income ≤ $10,000 :- 10%⟩, ⟨ | nb_children ≥ 3 :- 15%⟩ | true :- 20%⟩

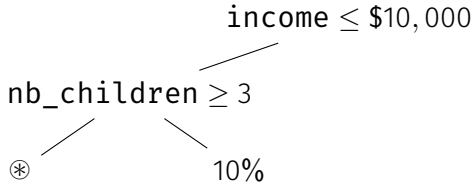
income = $9,000; nb_children = 4



Concolic execution of default terms

⟨⟨ | income ≤ $10,000 :- 10%⟩, ⟨ | nb_children ≥ 3 :- 15%⟩ | true :- 20%⟩

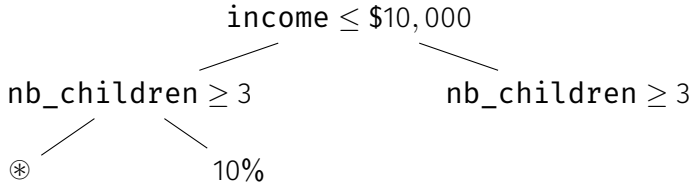
income = $9,000; nb_children = 2



Concolic execution of default terms

$\langle\langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \mid \texttt{true} \texttt{ :- } 20\% \rangle$

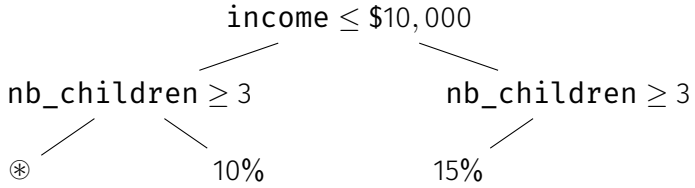
`income` = **$11,000**; `nb_children` = **4**



Concolic execution of default terms

⟨⟨ | income ≤ $10,000 :- 10%⟩, ⟨ | nb_children ≥ 3 :- 15%⟩ | true :- 20%⟩

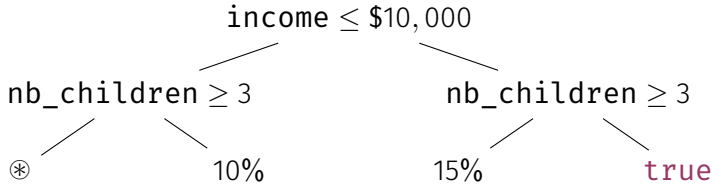
income = $11,000; nb_children = 4



Concolic execution of default terms

$\langle\langle \mid \texttt{income} \leq \$10,000 \texttt{ :- } 10\% \rangle, \langle \mid \texttt{nb\_children} \geq 3 \texttt{ :- } 15\% \rangle \mid \texttt{true} \texttt{ :- } 20\% \rangle$

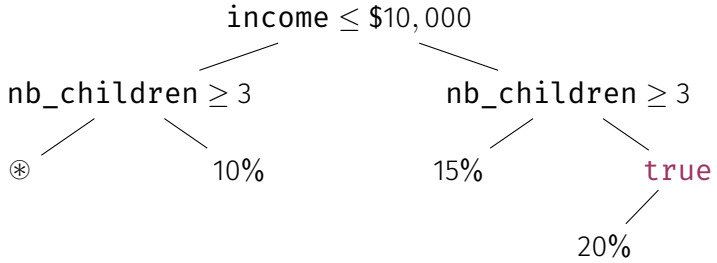
$\texttt{income} = \$11{,}000; \quad \texttt{nb\_children} = 2$



Concolic execution of default terms

⟨⟨ | income ≤ $10,000 :- 10%⟩, ⟨ | nb_children ≥ 3 :- 15%⟩ | true :- 20%⟩

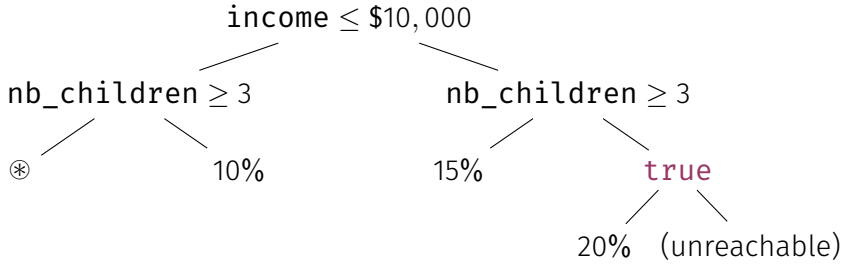
income = $11,000; nb_children = 2



Concolic execution of default terms

⟨⟨ | income ≤ $10,000 :- 10%⟩, ⟨ | nb_children ≥ 3 :- 15%⟩ | true :- 20%⟩

income = ???; nb_children = ???



Symbolic vs Concolic execution

Why not purely symbolic execution?

Symbolic vs Concolic execution

Why not purely symbolic execution?

- Possible incompleteness due to mixed integer/rational reasoning

Symbolic vs Concolic execution

Why not purely symbolic execution?

- Possible incompleteness due to mixed integer/rational reasoning
- Lists and dates can be hard to encode

Performance optimizations using reordering

Theorem (Independence of exception evaluation order)

If there is a default value v such that

$$\langle \ldots, e_i, \ldots, \mathbf{e_j}, \ldots \mid b_{default} \text{ :- } e_{default} \rangle \longrightarrow^* v,$$

then

$$\langle \ldots, \mathbf{e_j}, \ldots, e_i, \ldots \mid b_{default} \text{ :- } e_{default} \rangle \longrightarrow^* v.$$

Performance optimizations using reordering

Theorem (Independence of exception evaluation order)

If there is a default value v such that

$$\langle \ldots, e_i, \ldots, \boldsymbol{e_j}, \ldots \mid b_{default} \ :- \ e_{default} \rangle \longrightarrow^* v,$$

then

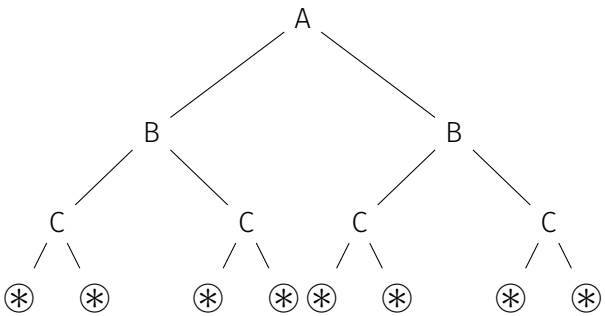
$$\langle \ldots, \boldsymbol{e_j}, \ldots, e_i, \ldots \mid b_{default} \ :- \ e_{default} \rangle \longrightarrow^* v.$$

Example:

$$\langle A, B, C, \circledast \mid b_{default} \ :- \ e_{default} \rangle \sim \langle \circledast, A, B, C \mid b_{default} \ :- \ e_{default} \rangle$$

Performance optimizations using reordering – Example

$$\langle A, B, C, \circledast \mid b_{default} :- e_{default} \rangle \sim \langle \circledast, A, B, C \mid b_{default} :- e_{default} \rangle$$



Usability improvement

```
# Article 3
If the income is less than $10,000, the percentage
mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

Usability improvement

```
# Article 3
If the income is less than $10,000, the percentage
mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

Query: income > $10,000 ?

Usability improvement

```
# Article 3
If the income is less than $10,000, the percentage
mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

Query: income > $10,000 ? → Answer: $10,000.01

Usability improvement

```
# Article 3
If the income is less than $10,000, the percentage
mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

Query: income > $10,000 ? $\rightarrow$ Answer: $10,000.01

- Difficult to compute by hand

Usability improvement

```
# Article 3
If the income is less than $10,000, the percentage
mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

Query: income > $10,000 ? $\rightarrow$ Answer: $10,000.01

- Difficult to compute by hand
- Find more usable input values using **soft constraints**

Usability improvement

```
# Article 3
If the income is less than $10,000, the percentage
mentioned at article 1 is 10%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.income <= $10,000
  consequence equals 10%
```
```

Query: income > $10,000 ? $\rightarrow$ Answer: ~~$10,000.01~~ **$11,000**

- Difficult to compute by hand
- Find more usable input values using soft constraints !
  - *e.g.* **round to $1,000**

Performance optimizations of the SMT solver

- Incremental mode

Performance optimizations of the SMT solver

- Incremental mode
  - Solver keeps a stack of constraints

Performance optimizations of the SMT solver

- Incremental mode
  - Solver keeps a stack of constraints
  - Keeps satisfiability status of save points

Performance optimizations of the SMT solver

- Incremental mode
  - Solver keeps a stack of constraints
  - Keeps satisfiability status of save points
  - Works best with depth-first exploration

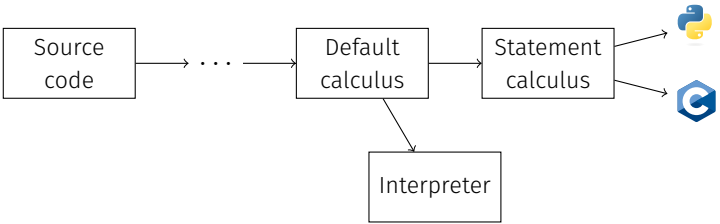
Performance optimizations of the SMT solver

- Incremental mode
  - Solver keeps a stack of constraints
  - Keeps satisfiability status of save points
  - Works best with depth-first exploration
  - Allows efficient soft constraints

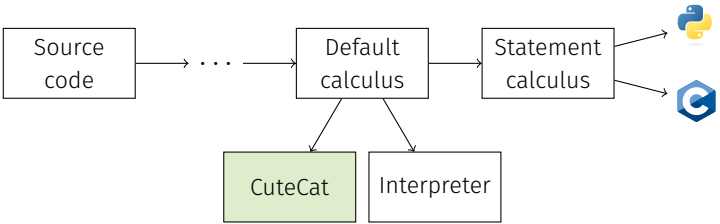
Performance optimizations of the SMT solver

- Incremental mode
  - Solver keeps a stack of constraints
  - Keeps satisfiability status of save points
  - Works best with depth-first exploration
  - Allows efficient soft constraints
- Redundant constraint elimination

Implementation of CuteCat



Implementation of CuteCat



- Integrated into Catala compiler's *default calculus* IR
- 3.4k lines of OCaml code
- **Z3** SMT Solver

Case study: housing benefits

French housing benefits: **5736** lines of law, **8615** lines of code

- 186,390 test cases generated in **7h of single-threaded CPU time**
- 366s spent in solver, the rest in evaluation
- Able to find a conflict: bedrooms vs flat sharing
- 99.83% of tests satisfy soft constraints

Overhead of the concolic execution

- 4.5x overhead w.r.t. Catala interpreter on our benchmarks
- SymCC [PF20] or SYMSAN [Che+22] reach the same order of magnitude
- KLEE sometimes reports several orders of magnitude [Yun+18]

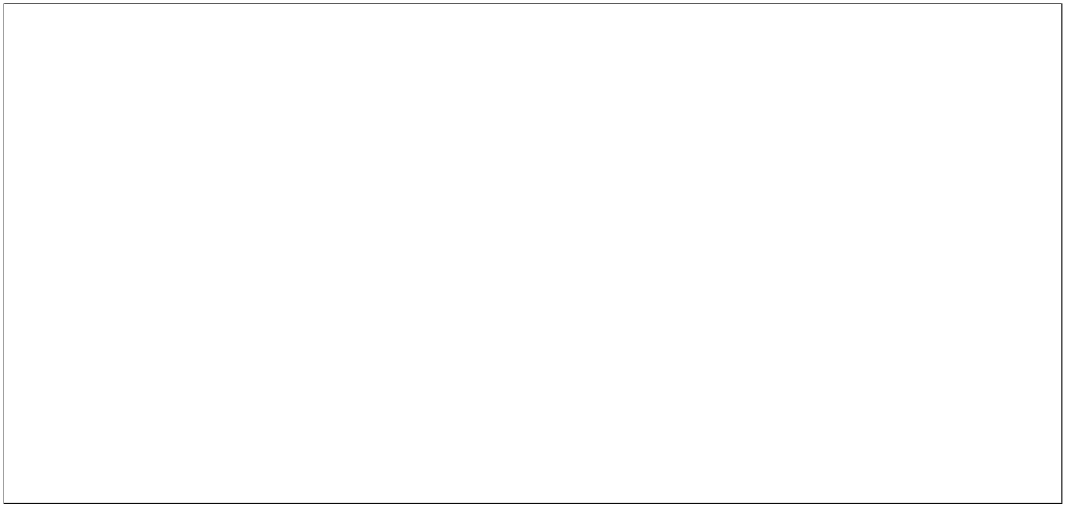
Overhead of the concolic execution

- 4.5x overhead w.r.t. Catala interpreter on our benchmarks
- SymCC [PF20] or SYMSAN [Che+22] reach the same order of magnitude
- KLEE sometimes reports several orders of magnitude [Yun+18]
- More optimizations in future work

Conclusion

Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]



Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]

Be aware of the specificities of implementing computational laws
#1-#3

Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]

Be aware of the specificities of implementing computational laws
#1-#3

Make your implementation abide by the law
#4-#7

Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]

Be aware of the specificities of implementing computational laws
#1-#3

Make your implementation abide by the law
#4-#7

Use the right programming tools
#8-#9

Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]

Be aware of the specificities of implementing computational laws
#1-#3

Make your implementation abide by the law
#4-#7

Use the right programming tools
#8-#9

Ease day-to-day technical development
#10-#14

Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]

Be aware of the specificities of implementing computational laws
#1-#3

Make your implementation abide by the law
#4-#7

Use the right programming tools
#8-#9

Ease day-to-day technical development
#10-#14

Ensure your project will last
#15-#17

Related Work within Catala Project – Recommendations

Recommendations for public administrations implementing laws [MH26]

Be aware of the specificities of implementing computational laws
#1-#3

Make your implementation abide by the law
#4-#7

Use the right programming tools
#8-#9

Ease day-to-day technical development
#10-#14

Ensure your project will last
#15-#17

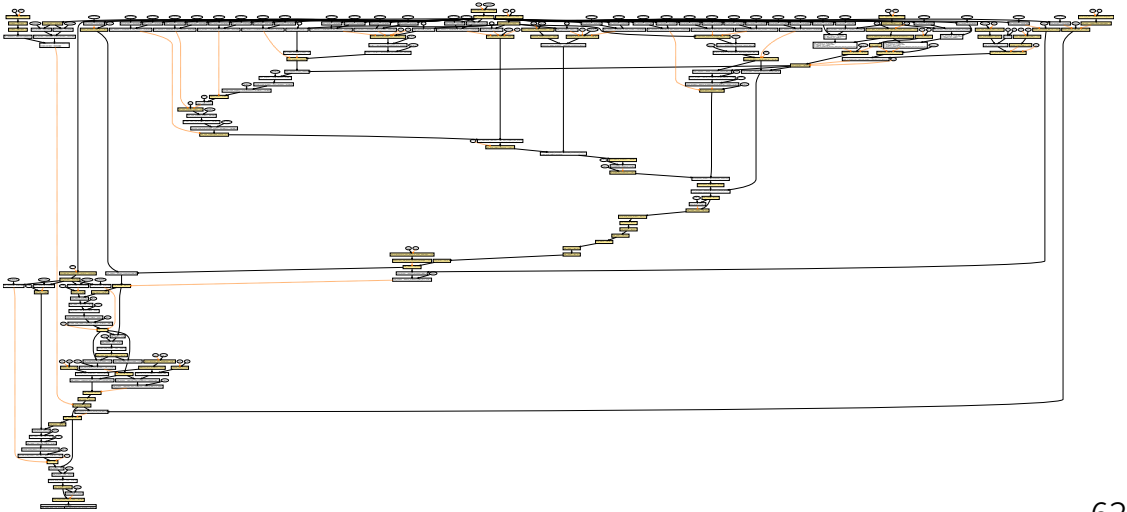
Facilitate public interaction
#18-#20

Related Work within Catala Project – Explainability of computations [Mer+24]

Explainability is a requirement mentioned by e.g. GDPR recital 71

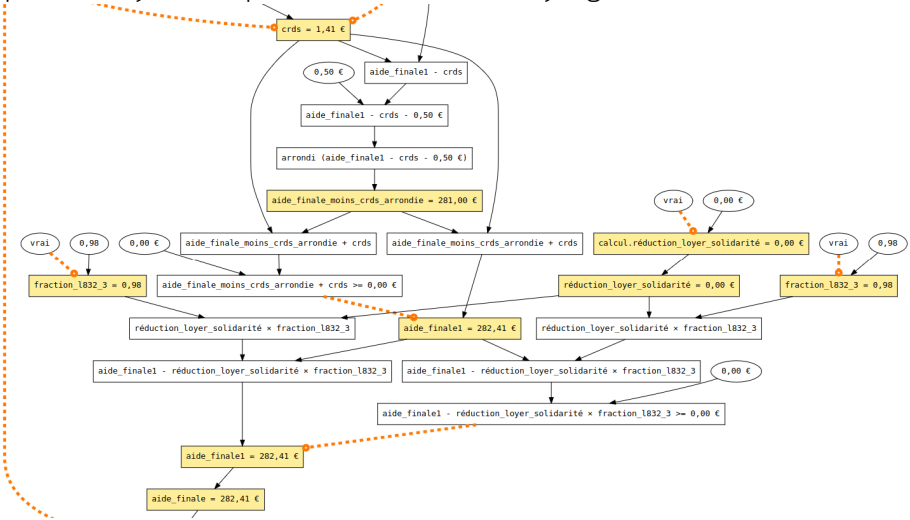
Related Work within Catala Project – Explainability of computations [Mer+24]

Explainability is a requirement mentioned by e.g. GDPR recital 71



Related Work within Catala Project – Explainability of computations [Mer+24]

Explainability is a requirement mentioned by e.g. GDPR recital 71



Related Work

- Date-related Computations

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]
- Auditing Government Benefits Screening Tools [EB20]

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]
- Auditing Government Benefits Screening Tools [EB20]
  - Pennsylvania's "Do I Qualify?"

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]
- Auditing Government Benefits Screening Tools [EB20]
  - Pennsylvania's "Do I Qualify?"
  - 68,983 generated households from demographic statistics, $\geq$ 4.6% errors.

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]
- Auditing Government Benefits Screening Tools [EB20]
  - Pennsylvania's "Do I Qualify?"
  - 68,983 generated households from demographic statistics, $\geq$ 4.6% errors.
  - NB: difficulties due to statistical confidentiality

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]
- Auditing Government Benefits Screening Tools [EB20]
  - Pennsylvania's "Do I Qualify?"
  - 68,983 generated households from demographic statistics, $\geq$ 4.6% errors.
  - NB: difficulties due to statistical confidentiality
- "Kill it with fire: Manage Aging computer systems" [Bel21]

Related Work

- Date-related Computations
  - Timezones, leap seconds, in Rocq [Ana+24]
  - Empirical study of date and time bugs in Python [Tiw+25]
  - SMT encoding of date arithmetic [CTP26]
- Auditing Government Benefits Screening Tools [EB20]
  - Pennsylvania's "Do I Qualify?"
  - 68,983 generated households from demographic statistics, $\geq$ 4.6% errors.
  - NB: difficulties due to statistical confidentiality
- "Kill it with fire: Manage Aging computer systems" [Bel21]
- "A Case for Feminism in Programming Language Design" [HS24]

Conclusion

- Legal software vs. *rocket science*

Conclusion

- Legal software vs. *rocket science*
- Potential for impactful, interdisciplinary research

Conclusion

- Legal software vs. *rocket science*
- Potential for impactful, interdisciplinary research
- Administrations and NGOs as a “third way” between academia and industry

References – I

[Ana+24] de Almeida Borges Ana et al. **“UTC Time, Formally Verified”.** In: ACM, 2024, pp. 2–13.

[Bal+19] Clément Ballabriga et al. **“Static Analysis of Binary Code with Memory Indirections Using Polyhedra”.** In: Lecture Notes in Computer Science. Springer, 2019, pp. 114–135.

[Bal19] Nathan Ballantyne. **“Epistemic trespassing”.** In: Mind 510 (2019), pp. 367–395.

[Bel21] Marianne Bellotti. **Kill it with fire: manage aging computer systems.** No Starch Press, 2021.

References – II

[Che+22] Ju Chen et al. “SYMSAN: Time and Space Efficient Concolic Execution via Dynamic Data-Flow Analysis”. In: USENIX Association, 2022, pp. 2531–2548.

[CTP26] Leyi Cui, Shrey Tiwari, and Rohan Padhye. DateSAT: A Framework for Solving Date and Period Constraints. 2026.

[DOM21] David Delmas, Abdelraouf Ouadjaout, and Antoine Miné. “Static Analysis of Endian Portability by Abstract Interpretation”. In: Lecture Notes in Computer Science. Springer, 2021, pp. 102–123.

References – III

[Dut20] Dutch Data Protection Authority. Werkwijze Belastingdienst in strijd met de wet en discriminerend. 2020. URL: https://autoriteitpersoonsgegevens.nl/actueel/werkwijze-belastingdienst-in-strijd-met-de-wet-en-discriminerend.

[EB20] Nel Escher and Nikola Banovic. "Exposing Error in Poverty Management Technology: A Method for Auditing Government Benefits Screening Tools". In: Proc. ACM Hum. Comput. Interact. CSCW (2020), 064:1–064:20.

[Esc+24] Nel Escher et al. "Code-ifying the Law: How Disciplinary Divides Afflict the Development of Legal Software". In: Proc. ACM Hum. Comput. Interact. CSCW2 (2024), pp. 1–37. DOI: 10.1145/3686937.

References – IV

[GFM25] Pierre Goutagny, Aymeric Fromherz, and Raphaël Monat. **“CUTECat: Concolic Execution for Computational Law”.** In: ed. by Viktor Vafeiadis. Lecture Notes in Computer Science. Springer, 2025, pp. 31–61. DOI: 10.1007/978-3-031-91121-7_2.

[HM22] Liane Huttner and Denis Merigoux. **“Catala: Moving Towards the Future of Legal Expert Systems”.** In: Artificial Intelligence and Law (Aug. 2022). DOI: 10.1007/s10506-022-09328-5.

[HS24] Felienne Hermans and Ari Schlesinger. **“A Case for Feminism in Programming Language Design”.** In: ed. by Jonathan Edwards and Marcel Taeumel. ACM, 2024, pp. 205–222. DOI: 10.1145/3689492.3689809.

References – V

[Law24] Sarah Lawsky. **“Computational Law and Epistemic Trespassing”.** In: Journal of Cross-disciplinary Research in Computational Law 2 (May 2024).

[MAS23] Denis Merigoux, Marie Alauzen, and Lilya Slimani. **“Rules, Computation and Politics. Scrutinizing Unnoticed Programming Choices in French Housing Benefits”.** In: Journal of Cross-disciplinary Research in Computational Law 1 (2023), p. 23.

[MCP21] Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko. **“Catala: a programming language for the law”.** In: 2021.

[Mer+24] Denis Merigoux et al. De la transparence à l’explicabilité automatisée des algorithmes : comprend Tech. rep. RR-9535. INRIA Paris, Jan. 2024, p. 68.

References – VI

[MFM24] Raphaël Monat, Aymeric Fromherz, and Denis Merigoux. **“Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law”.** In: ed. by Stephanie Weirich. Lecture Notes in Computer Science. Springer, 2024, pp. 421–450. DOI: 10.1007/978-3-031-57267-8_16.

[MH26] Raphaël Monat and Liane Huttner. **“Coding computational laws: 20 recommendations for public administrations”.** In: Information & Communications Technology Law 0 (2026), pp. 1–17. DOI: 10.1080/13600834.2026.2628389.

References – VII

[MMP21] Denis Merigoux, Raphaël Monat, and Jonathan Protzenko. **“A modern compiler for the French tax code”.** In: ed. by Aaron Smith, Delphine Demange, and Rajiv Gupta. ACM, 2021, pp. 71–82. DOI: 10.1145/3446804.3446850.

[Moc18] Percy Mockler. The Phoenix Pay Problem: Working toward a Solution. Senate Canada, 2018.

[PF20] Sebastian Poeplau and Aurélien Francillon. **“Symbolic Execution with SymCC: Don’t Interpret, Compile!”** In: SEC’20. USA: USENIX Association, 2020, pp. 181–198.

[Red+22] J Redden et al. **“Automating public services: learning from cancelled systems”.** In: Cardiff, UK: Data Justice Lab, Cardiff University (2022).

References – VIII

[Tiw+25] Shrey Tiwari et al. **“It’s About Time: An Empirical Study of Date and Time Bugs in Open-Source Python Software”.** In: 2025, pp. 39–51. DOI: 10.1109/MSR66628.2025.00020.

[Yun+18] Insu Yun et al. **“QSYM : A Practical Concolic Execution Engine Tailored for Hybrid Fuzzing”.** In: USENIX Association, 2018, pp. 745–761.