Automatic Verification for Computational Law

Pierre Goutagny[1] Aymeric Fromherz[2] Raphaël Monat[1]

Marktoberdorf Summer School, 12 August 2026

[1]Inria Lille, [2]Inria Paris

Computational law

- **Computational laws** specify algorithms: taxes, social benefits, etc.
- Administrations implement them as programs
- Critical: *e.g.* French military payroll system Louvois: 120k military personnel over- or under-paid, overpayments totalling 545M € to pay back

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

The Catala domain-specific language

```
# 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%
```
```

- Literate programming

```
# 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%
```
```

The Catala domain-specific language

```
# 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%
```
```

- Literate programming

```
# 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%
```
```

The Catala domain-specific language

```
# 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%
```
```

- Literate programming
- Follows the exception/default structure of the law

Kinds of error

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

Kinds of error

```
# 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 the code
  - interpretation conflicts, *e.g.* `income = $9,000` and `children = 4`
  - unhandled cases
  - in Catala: ambiguity = runtime error
  - resolved by lawyers/administration if implementation is correct

Kinds of error

```
# 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 the code
  - interpretation conflicts, *e.g.* `income = $9,000` and `children = 4`
  - unhandled cases
  - in Catala: ambiguity = runtime error
  - resolved by lawyers/administration if implementation is correct
- Other errors: division by zero, assertion error, etc.

Finding errors

What properties do we want in our search for errors?

- Find errors automatically
- Systematically:
  - find complex corner cases
  - complete coverage
- Handle default logic
- Generate (counter-)examples for non-programmer users

Finding errors

What properties do we want in our search for errors?

- Find errors automatically
- Systematically:
  - find complex corner cases
  - complete coverage
- Handle default logic
- Generate (counter-)examples for non-programmer users

→ CUTECat, a **concolic** (*conc*rete+symb*olic*) execution engine for Catala (ESOP'25)

Finding errors

What properties do we want in our search for errors?

- Find errors automatically
- Systematically:
  - find complex corner cases
  - complete coverage
- Handle default logic
- Generate (counter-)examples for non-programmer users

→ CUTECat, a **concolic** (*conc*rete+symb*olic*) execution engine for Catala (ESOP'25)

- Avoid some common obstacles for free:
  - no loops or memory
  - all programs terminate

Finding errors

What properties do we want in our search for errors?

- Find errors automatically
- Systematically:
  - find complex corner cases
  - complete coverage
- Handle default logic
- Generate (counter-)examples for non-programmer users

→ CUTECat, a **concolic** (*conc*rete+symb*olic*) execution engine for Catala (ESOP'25)

- Avoid some common obstacles for free:
  - no loops or memory
  - all programs terminate
- ∼**200k test cases** in less than **7h** on real-world example

Finding errors

What properties do we want in our search for errors?

- Find errors automatically
- Systematically:
  - find complex corner cases
  - complete coverage
- Handle default logic
- Generate (counter-)examples for non-programmer users

→ CUTECat, a **concolic** (*conc*rete+symb*olic*) execution engine for Catala (ESOP'25)

- Avoid some common obstacles for free:
  - no loops or memory
  - all programs terminate
- ∼**200k test cases** in less than **7h** on real-world example
- But some features are hard to encode symbolically

Future work

- Incremental analysis *e.g.* for amendments
- Conformance testing *e.g.* for simulators
- Handle complex features *e.g.* dates
- Improve user-friendliness for non-programmers

Future work

- Incremental analysis *e.g.* for amendments
- Conformance testing *e.g.* for simulators
- Handle complex features *e.g.* dates
- Improve user-friendliness for non-programmers

Contact, slides: pierregoutagny.fr