CUTECat

Concolic Execution for Computational Law

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

GT SISE, January 23 2025

[1]Inria Lille, [2]Inria Paris

Introduction

Computational law

- **Computational laws** encode 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
~~~

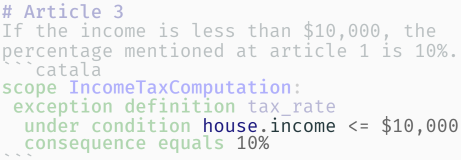
# Article 2
The fixed percentage mentioned at
article 1 is 20%.
```catala
scope IncomeTaxComputation:
  definition tax_rate equals 20%
```
```

```
# 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%
```

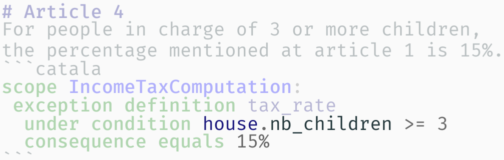
# Article 4
For people in charge of 3 or more
children, the percentage mentioned at
article 1 is 15%.
```catala
scope IncomeTaxComputation:
 exception definition tax_rate
  under condition house.nb_children >= 3
  consequence equals 15%
```
```

- 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 lawyer/court 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 lawyer/court 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-expert 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-expert users

→ Symbolic execution

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-expert users

→ Symbolic execution

- 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-expert users

→ Symbolic execution

- Avoid some common obstacles for free:
  - no loops or memory
  - all programs terminate
- But some features are hard to encode symbolically

Outline

Concolic execution of default terms

Performance and usability improvements

Experimental evaluation

Concolic execution of default terms

Concolic execution: first example

Concolic = *conc*rete + symb*olic*

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

Concolic execution: first example

Concolic = *conc*rete + symb*olic*

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

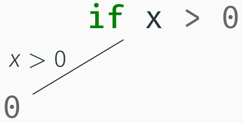
```
if x > 0
```

Step	x	y	Output	Constraints after evaluation	Next path to try

Concolic execution: first example

Concolic = *conc*rete + symb*olic*

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

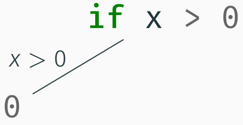


Step	x	y	Output	Constraints after evaluation	Next path to try
1	1	20	0	$x > 0$	

Concolic execution: first example

Concolic = *conc*rete + sym*bolic*

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



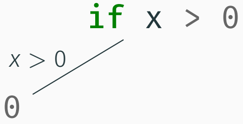
Step	x	y	Output	Constraints after evaluation	Next path to try
1	1	20	0	$x > 0$	$\neg(x > 0)$

⤸ SMT

Concolic execution: first example

Concolic = *concrete* + symbo*lic*

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



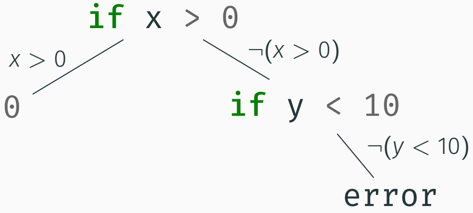
Step	x	y	Output	Constraints after evaluation	Next path to try
1	1	20	0	$x > 0$	$\neg(x > 0)$
2	0	20			

⤸ SMT

Concolic execution: first example

Conc*olic* = *conc*rete + symb*olic*

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



`if x > 0`

$x > 0$

`0`

$\neg(x > 0)$

`if y < 10`

$\neg(y < 10)$

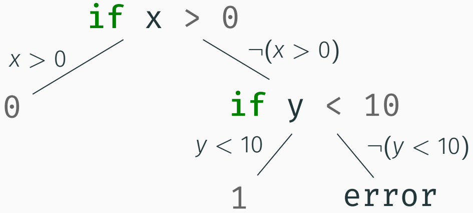
`error`

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

Concolic = *concrete* + symb*olic*

```
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	`error`	$\neg(x > 0) \wedge \neg(y < 10)$	$\neg(x > 0) \wedge y < 10$
3	0	9	1	$\neg(x > 0) \wedge y < 10$	-

⤸ SMT

⤸ SMT

Default terms: syntax

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

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

Default terms: syntax

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

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

Default terms: syntax

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

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

Default terms: syntax

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

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

Default terms: syntax

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

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

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

Default terms: semantics

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

Default terms: semantics

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

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 $b_{default} =$ `true`, then **evaluate** $e_{default}$
   - Else if $b_{default} =$ `false`, then **return** $\varnothing$

Default terms: semantics

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

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 $b_{default} =$ `true`, then **evaluate** $e_{default}$
   - Else if $b_{default} =$ `false`, then **return** $\varnothing$

Default terms: semantics

```
⟨
  ⟨ | income ≤ $10,000 :- 10%⟩,
  ⟨ | nb_children ≥ 3 :- 15%⟩
  | true :- 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 $b_{default}$ = `true`, then **evaluate** $e_{default}$
   - Else if $b_{default}$ = `false`, then **return** $\varnothing$

Default terms: semantics

```
⟨
  ⟨ | income ≤ $10,000 :- 10%⟩,
  ⟨ | nb_children ≥ 3 :- 15%⟩
  | true :- 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 $b_{default}$ = `true`, then **evaluate** $e_{default}$
   - Else if $b_{default}$ = `false`, then **return** $\varnothing$

Default terms: semantics

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

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** $\varnothing$

Concolic execution of default terms

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

Concolic execution of default terms

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

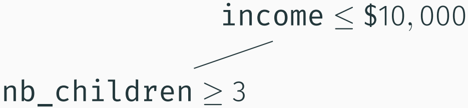
$\texttt{income}$ = \$9,000; $\texttt{nb\_children}$ = 4

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

Concolic execution of default terms

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

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



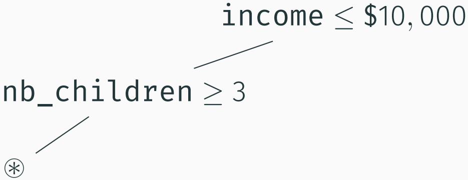
$\texttt{income} \leq \$10,000$

$\texttt{nb\_children} \geq 3$

Concolic execution of default terms

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

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

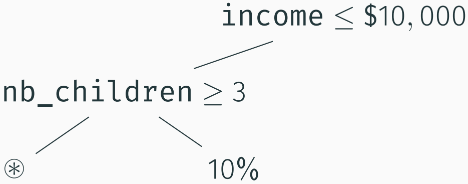


$\texttt{income} \leq \$10,000$
$\texttt{nb\_children} \geq 3$
$\circledast$

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; $\texttt{nb\_children}$ = 2



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

$\texttt{nb\_children} \geq 3$

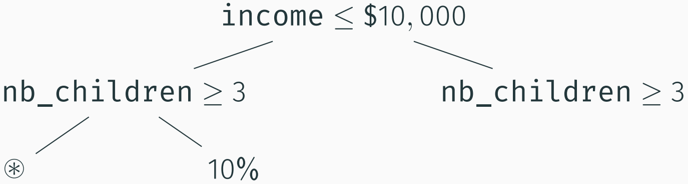
$\circledast$

10%

Concolic execution of default terms

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

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



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

$\texttt{nb\_children} \geq 3$

$\texttt{nb\_children} \geq 3$

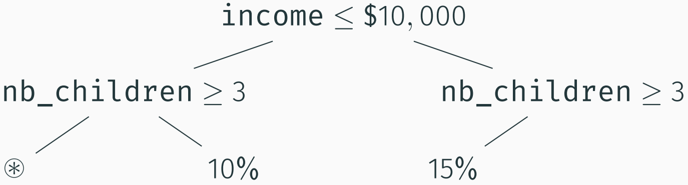
$\circledast$

10%

Concolic execution of default terms

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

$\texttt{income} = \$11{,}000;\quad \texttt{nb\_children} = 4$



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

$\texttt{nb\_children} \geq 3$

$\texttt{nb\_children} \geq 3$

$\circledast$

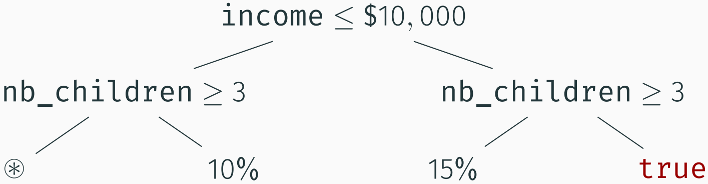
10%

15%

Concolic execution of default terms

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

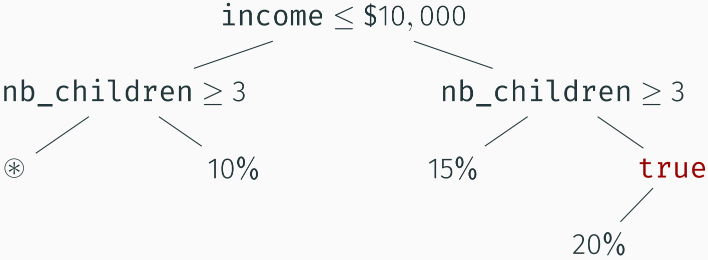
$\texttt{income} = \$11{,}000; \quad \texttt{nb\_children} = 2$



Concolic execution of default terms

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

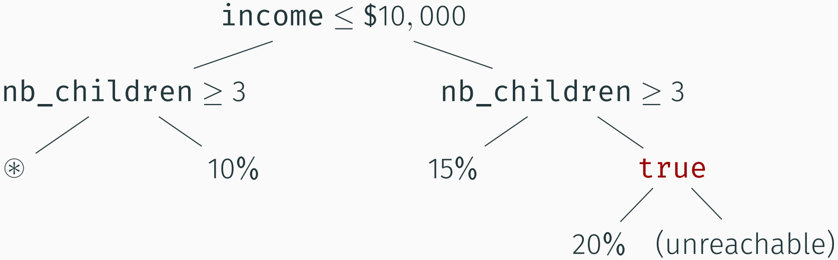
$\texttt{income} = \$11{,}000; \quad \texttt{nb\_children} = 2$



Concolic execution of default terms

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

$\texttt{income} = ???;$ $\qquad$ $\texttt{nb\_children} = ???$



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

$\texttt{nb\_children} \geq 3$

$\texttt{nb\_children} \geq 3$

$\circledast$

10%

15%

true

20%

(unreachable)

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.
→ it becomes an exception to the exception.

$$\left\langle \begin{array}{l} \langle\ |\ \texttt{income} \le \$10,000\ \texttt{:-}\ 10\%\rangle, \\ \langle\ |\ \texttt{nb\_children} \ge 3\ \texttt{:-}\ 15\%\rangle \\ |\ \texttt{true}\ \texttt{:-}\ 20\% \end{array}\right\rangle$$

$$\left\langle \begin{array}{l} \left\langle \begin{array}{l} \langle\ |\ \texttt{income} \le \$10,000\ \texttt{:-}\ 10\%\rangle \\ |\ \texttt{nb\_children} \ge 3\ \texttt{:-}\ 15\% \end{array}\right\rangle \\ |\ \texttt{true}\ \texttt{:-}\ 20\% \end{array}\right\rangle$$

Performance and usability improvements

Performance optimizations using reordering

Theorem (Independence of exception evaluation order)
If there is a default value v such that

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

then

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

Performance optimizations using reordering

Theorem (Independence of exception evaluation order)
If there is a default value v such that

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

then

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

Example:

$$\langle \ldots, \circledast \mid b_{default} :- e_{default} \rangle \sim \langle \circledast, \ldots \mid b_{default} :- e_{default} \rangle$$

Human-compatible test cases

~~~
# 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%
```
~~~

Human-compatible test cases

~~~
# 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 ?

Human-compatible test cases

~~~
# 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

Human-compatible test cases

~~~
# 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

- Difficult for lawyers to compute by hand

Human-compatible test cases

~~~~
# 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

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

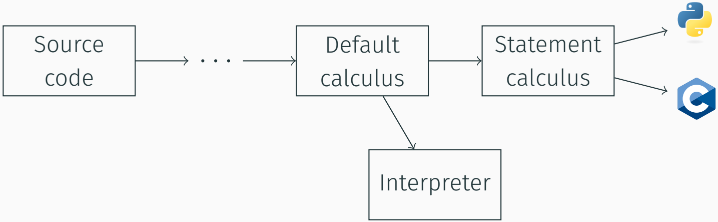
Human-compatible test cases

~~~~
# 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%
```
~~~~

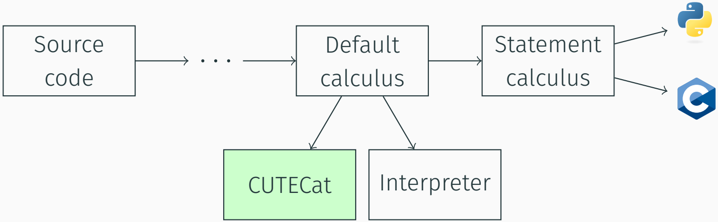
Query: income > $10,000 ? → Answer: ~~$10,000.01~~ **$11,000**

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

Implementation of CUTECat



Implementation of CUTECat



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

Experimental evaluation

Code base

Law	Lines of law in Markdown	Lines of Catala	Total
French housing benefits	5736	8615	14351

Code base

Law	Lines of law in Markdown	Lines of Catala	Total
French housing benefits	**5736**	**8615**	**14351**
US Tax code § 132	35	56	91
Minimum wage	74	161	235
Family quotient	36	165	201
Handwritten unit tests	139	699	838

Performance on small programs

Law	Time (s)		
	No optimizations	Optimized	Generated tests
US Tax code	0.27	0.02	10
Minimum wage	1.01	0.08	17
Family quotient	82.61	4.34	381

Case study: housing benefits

Key results

- 186,390 test cases generated in **7h of CPU time**
- 99.83% of tests satisfy soft constraints
- 366s spent in solver, the rest in evaluation
- 4.5x overhead w.r.t. Catala interpreter
- Able to find a conflict

Conclusion

Conclusion

- CUTECat: a concolic testing engine for computational law
- Novel concolic semantics for default logic
- Integrated with Catala toolchain
- Optimizations improve efficiency and usability by lawyers
- **186,390 test cases** in less than **7h** on real-world example

Conclusion

- CUTECat: a concolic testing engine for computational law
- Novel concolic semantics for default logic
- Integrated with Catala toolchain
- Optimizations improve efficiency and usability by lawyers
- **186,390 test cases** in less than **7h** on real-world example

Future work:

- Complex cases *e.g.* lists and dates
- Improve user-friendliness for non-technical users

Conclusion

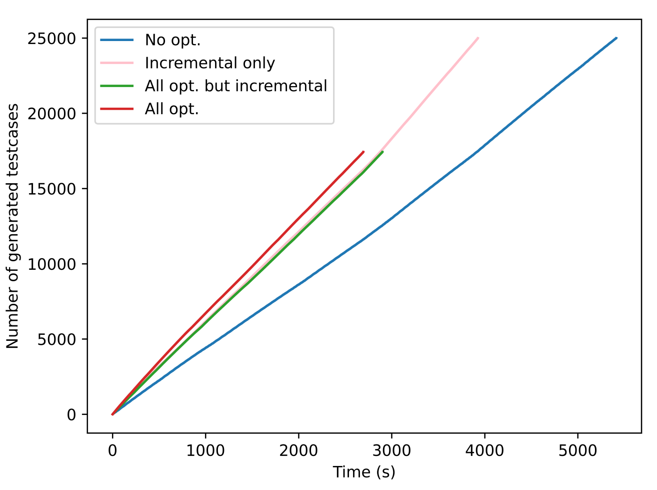
- CUTECat: a concolic testing engine for computational law
- Novel concolic semantics for default logic
- Integrated with Catala toolchain
- Optimizations improve efficiency and usability by lawyers
- **186,390 test cases** in less than **7h** on real-world example

Future work:

- Complex cases *e.g.* lists and dates
- Improve user-friendliness for non-technical users

Contact, preprint, slides: pierregoutagny.fr

Ablation study



Generated tests vs time