Catala: A Programming Language for the Law

Denis Merigoux, Nicolas Chataing, Jonathan Protzenko

Inria, Microsoft Research

April 27th, 2022

Law and Algorithms

US Tax Code, Section 121

(a) Exclusion
Gross income shall not include gain from the sale or exchange of property if, during the 5-year period ending on the date of the sale or exchange, such property has been owned and used by the taxpayer as the taxpayer's principal residence for periods aggregating 2 years or more.

(b) Limitations — (1) In general
The amount of gain excluded from gross income under subsection (a) with respect to any sale or exchange shall not exceed $250,000.

Law and Algorithms

US Tax Code, Section 121

(a) Exclusion

Gross income shall not include gain from the sale or exchange of property if, during the 5-year period ending on the date of the sale or exchange, such property has been owned and used by the taxpayer as the taxpayer's principal residence for periods aggregating 2 years or more.

(b) Limitations — (1) In general

The amount of gain excluded from gross income under subsection (a) with respect to any sale or exchange shall not exceed $250,000.

Scope of this work

Law and Algorithms

US Tax Code, Section 121

(a) Exclusion

Gross income shall not include gain from the sale or exchange of property if, during the 5-year period ending on the date of the sale or exchange, such property has been owned and used by the taxpayer as the taxpayer's principal residence for periods aggregating 2 years or more.

(b) Limitations — (1) In general

The amount of gain excluded from gross income under subsection (a) with respect to any sale or exchange shall not exceed $250,000.

Scope of this work

- Decision without human intervention
- No ambiguity in the decision procedure
- Quantitative data

Turning Law into (Pseudo-)Code

```
INCOME_EXCLUSION          =
  if

  ...

  then GAIN_FROM_SALE_OR_EXCHANGE
  else $0
```

"*Gross income shall not include gain from the sale or exchange of property if, [...]*"

Turning Law into (Pseudo-)Code

```
INCOME_EXCLUSION          =
  if
          ... truncate(PERIODS_OWNED, DATE_SALE - 5 years) ...
  and
          ... truncate(PERIODS_USED, DATE_SALE - 5 years) ...
  then GAIN_FROM_SALE_OR_EXCHANGE
  else $0
```

"*[...] during the 5-year period ending on the date of the sale or exchange, such property has been owned and used by the taxpayer as the taxpayer's principal residence [...]*"

Turning Law into (Pseudo-)Code

```
INCOME_EXCLUSION          =
  if
    aggregate(truncate(PERIODS_OWNED, DATE_SALE - 5 years)) > 2 years
  and
    aggregate(truncate(PERIODS_USED, DATE_SALE - 5 years)) > 2 years
  then GAIN_FROM_SALE_OR_EXCHANGE
  else $0
```

"*[...] for periods aggregating 2 years or more.*"

Turning Law into (Pseudo-)Code

```
INCOME_EXCLUSION_UNCAPPED =
  if
    aggregate(truncate(PERIODS_OWNED, DATE_SALE - 5 years)) > 2 years
  and
    aggregate(truncate(PERIODS_USED, DATE_SALE - 5 years)) > 2 years
  then GAIN_FROM_SALE_OR_EXCHANGE
  else $0

INCOME_EXCLUSION =
  if INCOME_EXCLUSION_UNCAPPED > $250,000
  then $250,000 else INCOME_EXCLUSION_UNCAPPED
```

“*The amount of gain excluded from gross income under subsection (a) with respect to any sale or exchange shall not exceed $250,000.*”

Public Legal Expert Systems: France

Name	Open-source	Language	Size (lines)
Income tax	open	M/C	100k
Housing tax	open	C	10k
Corporate tax	open	Java	10k
Payroll taxes	open	SQL	20k
Social benefits	open	COBOL	6,9M
Unemployment benefits	open	Java	1,3M
Pensions	closed	?	?
Estate tax	closed	?	?

Public Legal Expert Systems: France

Name	Open-source	Language	Size (lines)
Income tax	open	M/C	100k
Housing tax	open	C	10k
Corporate tax	open	Java	10k
Payroll taxes	open	SQL	20k
Social benefits	open	COBOL	6,9M
Unemployment benefits	open	Java	1,3M
Pensions	closed	?	?
Estate tax	closed	?	?

Not better in the US: IRS' income tax in 60's assembler!

Industrial Failures: Costly and Impactful

Louvois/SourceSolde

French army payroll software, very complicated rules for bonuses (174 of them).
⇒ **2011-2013: catastrophic failures, penniless soldiers**
⇒ **465M€ computation errors in 2012**
⇒ **Project cost in 2018: 80M€ + 156M€.**

Industrial Failures: Costly and Impactful

Louvois/SourceSolde

French army payroll software, very complicated rules for bonuses (174 of them).
⇒ **2011-2013: catastrophic failures, penniless soldiers**
⇒ **465M€ computation errors in 2012**
⇒ **Project cost in 2018: 80M€ + 156M€.**

Economic Impact Payment

US stimulus checks distributed to all US citizens.
⇒ **Almost 1M received incorrect EIP in 2020**
⇒ **10k ineligible due to commercial tax software error**
⇒ **10k military ineligible due to programming expired data error**

Code Validation: Current State of the Art

Test cases

⇒ **Common under-testing:**

- Thousands of cases required in a typical legal expert system
- Complete review and update necessary after legislative change
- Written by lawyers, costly

Code Validation: Current State of the Art

Test cases

⇒ **Common under-testing:**

- Thousands of cases required in a typical legal expert system
- Complete review and update necessary after legislative change
- Written by lawyers, costly

Need of a better communication medium for lawyers and programmers

Bringing Code and Law Together

Highlight of this work

Let's mix code and law in a single document and make it easy for lawyers and programmers to work on it.
⇒ **Domain-specific Language for the Law**

Bringing Code and Law Together

Highlight of this work

Let's mix code and law in a single document and make it easy for lawyers and programmers to work on it.
⇒ **Domain-specific Language for the Law**

Three major contributions:

1. Domain-specific language usable and reviewable by lawyers
2. Formalized semantics for legal reasoning
3. Partially certified compiler

❶ Catala: A Language Reviewable by lawyers

US Tax Code, Section 132, (c)(1) Qualified employee discount

The term “qualified employee discount” means any employee discount with respect to qualified property or services to the extent such discount does not exceed—
(A) in the case of property, the gross profit percentage of the price at which the property is being offered by the employer to customers

❶ Catala: A Language Reviewable by lawyers

US Tax Code, Section 132, (c)(1) Qualified employee discount

The term “qualified employee discount” means any employee discount with respect to qualified property or services to the extent such discount does not exceed—
(A) in the case of property, the gross profit percentage of the price at which the property is being offered by the employer to customers

```
scope QualifiedEmployeeDiscount :
  definition qualified_employee_discount
    under condition is_property consequence equals
      if employee_discount >$ customer_price *$ gross_profit_percentage then
        customer_price *$ gross_profit_percentage
      else employee_discount
```

❶ Catala: A Language Reviewable by lawyers

US Tax Code, Section 132, (c)(1) Qualified employee discount

(B) in the case of services, 20 percent of the price at which the services are being offered by the employer to customers.

❶ Catala: A Language Reviewable by lawyers

US Tax Code, Section 132, (c)(1) Qualified employee discount

(B) in the case of services, 20 percent of the price at which the services are being offered by the employer to customers.

```
scope QualifiedEmployeeDiscount :
  definition qualified_employee_discount
    under condition is_services consequence equals
      if employee_discount >$ customer_price *$ 20% then
        customer_price *$ 20%
      else employee_discount
```

The Non-Monotonic Legal Reasoning [Lawsky 2017, 2018]

> ***(A) $500,000 Limitation for certain joint returns*** *Paragraph (1) shall be applied by substituting "$500,000" for "$250,000" if— (i) either spouse meets the ownership requirements of subsection (a) with respect to such property; (ii) both spouses meet the use requirements of subsection (a) with respect to such property; and (iii) neither spouse is ineligible for the benefits of subsection (a) with respect to such property by reason of paragraph (3).*
> ***(B) Other joint returns*** *If such spouses do not meet the requirements of subparagraph (A), the limitation under paragraph (1) shall be the sum of the limitations under paragraph (1) to which each spouse would be entitled if such spouses had not been married. For purposes of the preceding sentence, each spouse shall be treated as owning the property during the period that either spouse owned the property.*

US Tax Code, Section 121, (b), (2)

The Non-Monotonic Legal Reasoning [Lawsky 2017, 2018]

(A) $500,000 Limitation for certain joint returns *Paragraph (1) shall be applied by substituting “$500,000” for “$250,000” if— (i) either spouse meets the ownership requirements of subsection (a) with respect to such property; (ii) both spouses meet the use requirements of subsection (a) with respect to such property; and (iii) neither spouse is ineligible for the benefits of subsection (a) with respect to such property by reason of paragraph (3).*
(B) Other joint returns *If such spouses do not meet the requirements of subparagraph (A), the limitation under paragraph (1) shall be the sum of the limitations under paragraph (1) to which each spouse would be entitled if such spouses had not been married. For purposes of the preceding sentence, each spouse shall be treated as owning the property during the period that either spouse owned the property.*

US Tax Code, Section 121, (b), (2)

[Base case: condition]

The Non-Monotonic Legal Reasoning [Lawsky 2017, 2018]

(A) $500,000 Limitation for certain joint returns *Paragraph (1) shall be applied by substituting "$500,000" for "$250,000" if— (i) either spouse meets the ownership requirements of subsection (a) with respect to such property; (ii) both spouses meet the use requirements of subsection (a) with respect to such property; and (iii) neither spouse is ineligible for the benefits of subsection (a) with respect to such property by reason of paragraph (3).*
(B) Other joint returns *If such spouses do not meet the requirements of subparagraph (A), the limitation under paragraph (1) shall be the sum of the limitations under paragraph (1) to which each spouse would be entitled if such spouses had not been married. For purposes of the preceding sentence, each spouse shall be treated as owning the property during the period that either spouse owned the property.*

US Tax Code, Section 121, (b), (2)

[Base case: condition] ⇒ [Base case: consequence]

The Non-Monotonic Legal Reasoning [Lawsky 2017, 2018]

(A) $500,000 Limitation for certain joint returns *Paragraph (1) shall be applied by substituting "$500,000" for "$250,000" if— (i) either spouse meets the ownership requirements of subsection (a) with respect to such property; (ii) both spouses meet the use requirements of subsection (a) with respect to such property; and (iii) neither spouse is ineligible for the benefits of subsection (a) with respect to such property by reason of paragraph (3).*
(B) Other joint returns *If such spouses do not meet the requirements of subparagraph (A), the limitation under paragraph (1) shall be the sum of the limitations under paragraph (1) to which each spouse would be entitled if such spouses had not been married. For purposes of the preceding sentence, each spouse shall be treated as owning the property during the period that either spouse owned the property.*

US Tax Code, Section 121, (b), (2)

[Base case: condition] ⇒ [Base case: consequence]
[Exception: condition]

The Non-Monotonic Legal Reasoning [Lawsky 2017, 2018]

(A) $500,000 Limitation for certain joint returns *Paragraph (1) shall be applied by substituting “$500,000” for “$250,000” if— (i) either spouse meets the ownership requirements of subsection (a) with respect to such property; (ii) both spouses meet the use requirements of subsection (a) with respect to such property; and (iii) neither spouse is ineligible for the benefits of subsection (a) with respect to such property by reason of paragraph (3).*
(B) Other joint returns *If such spouses do not meet the requirements of subparagraph (A), the limitation under paragraph (1) shall be the sum of the limitations under paragraph (1) to which each spouse would be entitled if such spouses had not been married. For purposes of the preceding sentence, each spouse shall be treated as owning the property during the period that either spouse owned the property.*

US Tax Code, Section 121, (b), (2)

[Base case: condition] ⇒ [Base case: consequence]
[Exception: condition] ⇒ [Exception: consequence]

The Core of Catala: Default Calculus

Behind the surface syntax, a formalized core!

The Core of Catala: Default Calculus

Behind the surface syntax, a formalized core!

$$
\begin{array}{llcll}
\text{Type} & \tau & ::= & \texttt{bool} \mid \texttt{unit} & \text{boolean and unit types} \\
 & & \mid & \tau \rightarrow \tau & \text{function type} \\
 & & & & \\
\text{Expression} & e & ::= & x \mid \texttt{true} \mid \texttt{false} \mid () & \text{variable, literal} \\
 & & \mid & \lambda\ (x : \tau) .\ e \mid e\ e & \lambda\text{-calculus}
\end{array}
$$

The Core of Catala: Default Calculus

Behind the surface syntax, a formalized core!

Type	τ	$::=$	$\texttt{bool} \mid \texttt{unit}$	boolean and unit types
		$\mid$	$\tau \rightarrow \tau$	function type
Expression	e	$::=$	$x \mid \texttt{true} \mid \texttt{false} \mid ()$	variable, literal
		$\mid$	$\lambda\ (x : \tau)\ .\ e \mid e\ e$	λ-calculus
		$\mid$	d	default term
Default	d	$::=$	$\langle e^* \mid e :- e \rangle$	default term
		$\mid$	$\circledast$	conflict error
		$\mid$	$\varnothing$	empty error

❷ Formalized semantics for legal reasoning

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false :- } e \rangle \longrightarrow \varnothing$$

❷ Formalized semantics for legal reasoning

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} \;\texttt{:-}\; e \rangle \longrightarrow \varnothing$$

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{true} \;\texttt{:-}\; v \rangle \longrightarrow v$$

❷ Formalized semantics for legal reasoning

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} \;\texttt{:-}\; e \rangle \longrightarrow \varnothing$$

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{true} \;\texttt{:-}\; v \rangle \longrightarrow v$$

$$\langle \varnothing, \ldots, \varnothing, v, \varnothing, \ldots, \varnothing \mid e_1 \;\texttt{:-}\; e_2 \rangle \longrightarrow v \qquad v \neq \varnothing$$

❷ Formalized semantics for legal reasoning

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{false} :- e \rangle \longrightarrow \varnothing$$

$$\langle \varnothing, \ldots, \varnothing \mid \texttt{true} :- v \rangle \longrightarrow v$$

$$\langle \varnothing, \ldots, \varnothing, v, \varnothing, \ldots, \varnothing \mid e_1 :- e_2 \rangle \longrightarrow v \qquad v \neq \varnothing$$

$$\frac{\text{nonempty\_count}(v_1, \ldots, v_n) > 1}{\langle v_1, \ldots, v_n \mid e_1 :- e_2 \rangle \longrightarrow \circledast}$$

Deploying Catala for Production

Interpreted ✗

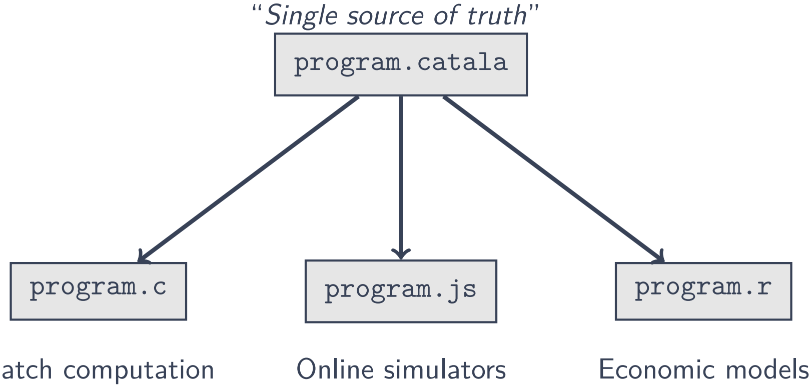
Deploying Catala for Production

Interpreted ✗

Compiled ✓

Deploying Catala for Production

Interpreted ✗ Compiled ✓



Compiling away the default term

$$\frac{e_1 \rightrightarrows e'_1 \qquad \cdots \qquad e_n \rightrightarrows e'_n \qquad e_{\text{just}} \rightrightarrows e'_{\text{just}} \qquad e_{\text{cons}} \rightrightarrows e'_{\text{cons}}}{\langle e_1 , \ldots , e_n \mid e_{\text{just}} :- e_{\text{cons}} \rangle \rightrightarrows}$$

Compiling away the default term

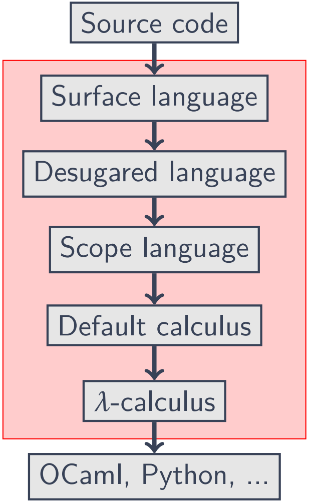
$$\frac{e_1 \Rrightarrow e_1' \qquad \cdots \qquad e_n \Rrightarrow e_n' \qquad e_{\text{just}} \Rrightarrow e_{\text{just}}' \qquad e_{\text{cons}} \Rrightarrow e_{\text{cons}}'}{\begin{array}{c}\langle e_1 \,, \ldots , e_n \mid e_{\text{just}} \;\texttt{:-}\; e_{\text{cons}} \rangle \Rrightarrow \\ \texttt{let}\ r_{\text{exceptions}} = \texttt{process\_exceptions}\,[\lambda\ \_ \rightarrow e_1' \,; \ldots ; \lambda\ \_ \rightarrow e_n'\,]\ \texttt{in} \\ \texttt{match}\ r_{\text{exceptions}}\ \texttt{with}\ \texttt{Some}\ e' \rightarrow e' \mid \texttt{None} \rightarrow \texttt{if}\ e_{\text{just}}'\ \texttt{then}\ e_{\text{cons}}'\ \texttt{else}\ \texttt{raise}\ \varnothing\end{array}}$$

Compiling away the default term

$$\frac{e_1 \Rrightarrow e_1' \qquad \cdots \qquad e_n \Rrightarrow e_n' \qquad e_{\text{just}} \Rrightarrow e_{\text{just}}' \qquad e_{\text{cons}} \Rrightarrow e_{\text{cons}}'}{\begin{array}{c} \langle e_1, \ldots, e_n \mid e_{\text{just}} :- e_{\text{cons}} \rangle \Rrightarrow \\ \texttt{let}\ r_{\text{exceptions}} = \texttt{process\_exceptions}\,[\lambda\,\_ \to e_1'; \ldots; \lambda\,\_ \to e_n']\ \texttt{in} \\ \texttt{match}\ r_{\text{exceptions}}\ \texttt{with}\ \texttt{Some}\ e' \to e' \mid \texttt{None} \to \texttt{if}\ e_{\text{just}}'\ \texttt{then}\ e_{\text{cons}}'\ \texttt{else}\ \texttt{raise}\ \varnothing \end{array}}$$

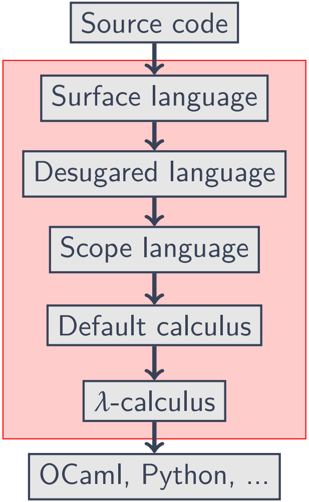
$$\begin{array}{lcl}
\texttt{process\_exceptions} & : & \texttt{list}(\texttt{unit} \to \tau) \to \texttt{option}\,\tau \\
\texttt{process\_exceptions} & \triangleq & \texttt{fold\_left}\,(\lambda\,(a : \texttt{option}\,\tau)\,(e' : \texttt{unit} \to \tau)\,. \\
 & & \quad \texttt{let}\ e' : \tau = \texttt{try}\ \texttt{Some}\ (e'())\ \texttt{with}\ \varnothing \to \texttt{None}\ \texttt{in} \\
 & & \quad \texttt{match}\ (a, e')\ \texttt{with} \\
 & & \qquad \mid (\texttt{None}, e') \to e' \\
 & & \qquad \mid (\texttt{Some}\ a, \texttt{None}) \to \texttt{Some}\ a \\
 & & \qquad \mid (\texttt{Some}\ a, \texttt{Some}\ e') \to \texttt{raise}\ \circledast)\ \texttt{None}
\end{array}$$

The Catala compiler



- **Multiple passes architecture**

The Catala compiler



- **Multiple passes architecture**
- 13 000 lines of OCaml
- Looking for contributors!
- `github.com/CatalaLang/Catala`

❸ Certifying the compilation of default terms

Simulation theorem:

$$\begin{tikzcd}
e_{\mathrm{d}} \arrow[r] \arrow[d, phantom, "\downdownarrows" description] & e'_{\mathrm{d}} \arrow[d, phantom, "\downdownarrows" description] \\
e_{\lambda} \arrow[dr, "*"] & e'_{\lambda} \arrow[d, "*"] \\
 & e_{\mathrm{target}}
\end{tikzcd}$$

✔ Proven in F* (3500 lines)

The F* certification theorem

```
module D = DefaultCalculus; module L = LambdaCalculus
val translation_correctness (de: D.exp) (dtau: D.ty) : Lemma
  (requires (D.typing D.empty de dtau)) (ensures (
    let le = translate_exp de in let ltau = translate_ty dtau in
    L.typing L.empty le ltau ∧ begin
      if D.is_value de then L.is_value le else begin
        D.progress de dtau; D.preservation de dtau;
        let de' = Some?.v (D.step de) in
        translation_preserves_empty_typ de dtau;
        translation_preserves_empty_typ de' dtau;
        let le' : typed_l_exp ltau = translate_exp de' in
        exists (n1 n2:ℕ) (target: typed_l_exp ltau).
          (take_l_steps ltau le n1 == Some target ∧
           take_l_steps ltau le' n2 == Some target) end end))
```

Removing exceptions

$\varnothing \quad \Rrightarrow \quad$ `None`

Removing exceptions

$$\begin{aligned} \emptyset &\quad \rightrightarrows \quad \texttt{None} \\ \langle e_1, \ldots, e_n \mid e_{\texttt{just}} \texttt{ :- } e_{\texttt{cons}} \rangle &\quad \rightrightarrows \end{aligned}$$

```
match handle_exceptions [e_1, ..., e_n] with
| None -> bind e_just ~f:(fun b -> if b then e_cons else None)
| Some v -> Some v
```

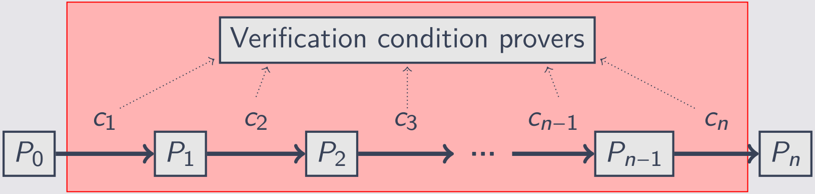
Future work: Catala as a proof platform

Properties we want to prove (with Alain Delaët, Aymeric Fromherz and Raphaël Monat):

- Well-behaved execution: absence of $\varnothing$, $\circledast$
- High-level functional properties: progressiveness of taxes, absence of loopholes, etc.

Proof by refinement

Syntax-directed certified program transformations:



Conclusion

Turning law into code is difficult; programmers and lawyers need better tooling

The Catala language – https://catala-lang.org

1. Domain-specific language usable and reviewable by lawyers
2. Formalized semantics for legal reasoning
3. Partially certified compiler

Future work: compilation to other languages (C/CUDA?), connection to proof backends

denis.merigoux@inria.fr