Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law

Raphaël Monat, Aymeric Fromherz, Denis Merigoux

Cambium seminar
29 March 2024
Some Catala News, and an Introduction
Legal implementations

Some legal implementations are critical software: taxes, benefits
Some legal implementations are critical software: taxes, benefits

Catala

- a DSL for computational laws
Legal implementations

Some legal implementations are **critical software**: taxes, benefits

Catala

- a DSL for computational laws
- providing transparency
Legal implementations

Some legal implementations are critical software: taxes, benefits

Catala

- a DSL for computational laws
- providing transparency
- easing maintenance
Legal implementations

Some legal implementations are **critical software**: taxes, benefits

<table>
<thead>
<tr>
<th>Catala</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>► a DSL for computational laws</td>
<td>► easing maintenance</td>
</tr>
<tr>
<td>► providing transparency</td>
<td>► through interdisciplinary work</td>
</tr>
</tbody>
</table>
News: Catala is getting closer to the real world

Partnerships with government agencies

DGFiP
Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more resources.

CNAF
Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.

Industrialization ongoing at Inria

- Recognized as a priority program of support to public policies
- From 1 to 3 research engineers in the coming months (+ Denis)
News: Catala is getting closer to the real world

Partnerships with government agencies

- DGFiP: Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more resources.
- CNAF: Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.

Industrialization ongoing at Inria

- Recognized as a priority program of support to public policies
- From 1 to 3 research engineers in the coming months (+ Denis)
News: Catala is getting closer to the real world

<table>
<thead>
<tr>
<th>Partnerships with government agencies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DGFiP</strong></td>
</tr>
<tr>
<td>Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more resources.</td>
</tr>
</tbody>
</table>
## News: Catala is getting closer to the real world

### Partnerships with government agencies

<table>
<thead>
<tr>
<th>Agency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGFiP</td>
<td>Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more ressources.</td>
</tr>
<tr>
<td>CNAF</td>
<td>Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.</td>
</tr>
</tbody>
</table>
News: Catala is getting closer to the real world

## Partnerships with government agencies

<table>
<thead>
<tr>
<th>Agency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGFiP</td>
<td>Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more resources.</td>
</tr>
<tr>
<td>CNAF</td>
<td>Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.</td>
</tr>
</tbody>
</table>
News: Catala is getting closer to the real world

### Partnerships with government agencies

<table>
<thead>
<tr>
<th>Agency</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DGFiP</strong></td>
<td>Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more resources.</td>
</tr>
<tr>
<td><strong>CNAF</strong></td>
<td>Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.</td>
</tr>
</tbody>
</table>

### Industrialization ongoing at Inria
**News: Catala is getting closer to the real world**

**Partnerships with government agencies**

<table>
<thead>
<tr>
<th>DGFiP</th>
<th>Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more resources.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNAF</td>
<td>Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.</td>
</tr>
</tbody>
</table>

**Industrialization ongoing at Inria**

- Recognized as a priority program of support to public policies
News: Catala is getting closer to the real world

Partnerships with government agencies

<table>
<thead>
<tr>
<th>Agency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGFiP</td>
<td>Since June 2023, re-implementing income tax with a real tax inspector. First results encouraging, asking for more ressources.</td>
</tr>
<tr>
<td>CNAF</td>
<td>Technical feasibility of Catala at CNAF (alongside COBOL and Oracle) confirmed, next step is launching an experimentation on a particular benefit.</td>
</tr>
</tbody>
</table>

Industrialization ongoing at Inria

- Recognized as a priority program of support to public policies
- From 1 to 3 research engineers in the coming months (+ Denis)
Handling Date Computations in Catala
Computing dates

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

2024-03-02
Computing dates

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

$ 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 date semantics

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
A wide variety of date semantics

Different legal bodies and choices

- 1 month = 30 days (Council of European Communities)
- When do leapers become adults?
A wide variety of date semantics

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 date semantics

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
### 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

⇒ Formal, flexible semantics required!
### 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

⇒ Formal, flexible semantics required! Focus on Gregorian calendar.
Outline

1. Semantics
2. Formalized Properties
3. Rounding-insensitivity Static Analysis
4. Case Study: French Housing Benefits
5. Conclusion
Semantics
values $v ::= (y, m, d) \mid \bot$

date unit $\delta ::= y \mid m \mid d$

expressions $e ::= v \mid e +_\delta n$
Semantics – Values

values $v ::= (y, m, d) | \bot$

date unit $\delta ::= y | m | d$

expressions $e ::= v | e +_\delta n$

$$nb\_days(y, m) = \begin{cases} 
29 \text{ if } m = 2 \land \text{is\_leap}(y) \\
28 \text{ if } m = 2 \land \neg\text{is\_leap}(y) \\
30 \text{ if } m \in \{ \text{Apr}, \text{Jun}, \text{Sep}, \text{Nov} \} \\
31 \text{ otherwise}
\end{cases}$$
Day additions with invalid day number propagate errors
Day additions with invalid day number propagate errors

\[
\text{ADD-DAYS-ERR1} \\
\begin{array}{c}
day < 1 \\
(y, m, day) + d n \rightarrow \bot
\end{array}
\]
Semantics – invalid dates

Day additions with invalid day number propagate errors

\[
\frac{\text{ADD-DAYS-ERR1}}{\text{day} < 1} \quad (y, m, day) +_d n \rightarrow \perp
\]

\[
\frac{\text{ADD-DAYS-ERR2}}{\text{day} > \text{nb_days}(y, m)} \quad (y, m, day) +_d n \rightarrow \perp
\]
Semantics – some cases of month addition

**Add-Month**

\[
1 \leq m + n \leq 12
\]

\[
(y, m, d) +_m n \rightarrow (y, m + n, d)
\]
Semantics – some cases of month addition

**Add-Month**

\[ 1 \leq m + n \leq 12 \]

\[ (y, m, d) +_m n \rightarrow (y, m + n, d) \]

**Add-Month-Over**

\[ m + n > 12 \]

\[ (y, m, d) +_m n \rightarrow (y + 1, m, d) +_m (n - 12) \]
Semantics – some cases of month addition

**Add-Month**

\[ 1 \leq m + n \leq 12 \]

\[ (y, m, d) + m n \rightarrow (y, m + n, d) \]

**Add-Month-Over**

\[ m + n > 12 \]

\[ (y, m, d) + m n \rightarrow (y + 1, m, d) + m (n - 12) \]

Similar cases for Add-Month-Under, year, day addition.
Semantics – Rounding

\[(2024, 01, 31) +_m 1 \rightarrow (2024, 02, 31)\]
Semantics – Rounding

(2024, 01, 31) +_m 1 \rightarrow (2024, 02, 31)

Rounding to valid dates required!
(2024, 01, 31) +_m 1 \rightarrow (2024, 02, 31)

Rounding to valid dates required!

\[
\text{rounding mode } \quad r \ ::= \quad \uparrow \mid \downarrow \mid \perp \\
\text{expressions} \quad \quad e \ ::= \quad v \mid e +\delta n \mid \text{rnd}_r e
\]
Semantics – Rounding

(2024, 01, 31) +_m 1 → (2024, 02, 31)

Rounding to valid dates required!

rounding mode \( r \) ::= ↑ | ↓ | ⊥

expressions \( e \) ::= v | e +_δ n | \( \text{rnd}_r \ e \)

\( \text{rnd}_↓(2024, 02, 31) = (2024, 02, 29) \)

\( \text{rnd}_↑(2024, 02, 31) = (2024, 03, 01) \)

Coreutils-like rounding not defined here
Semantics – Rounding

\[(2024, 01, 31) +_m 1 \rightarrow (2024, 02, 31)\]

Rounding to valid dates required!

rounding mode \( r ::= \uparrow | \downarrow | \perp \)

expressions \( e ::= v | e +_\delta n | \text{rnd}_r e \)

\( \text{rnd}_\uparrow(2024, 02, 31) = (2024, 03, 01) \)
\( \text{rnd}_\downarrow(2024, 02, 31) = (2024, 02, 29) \)
Semantics – Rounding

(2024, 01, 31) + _m_ 1 \rightarrow (2024, 02, 31)

Rounding to valid dates required!

rounding mode \quad r ::= \uparrow \mid \downarrow \mid \bot

expressions \quad e ::= v \mid e + \delta n \mid \text{rnd}_r \ e

\text{rnd}_\uparrow(2024, 02, 31) = (2024, 03, 01)
\text{rnd}_\downarrow(2024, 02, 31) = (2024, 02, 29)
\text{rnd}_\bot(2024, 02, 31) = \bot
Semantics – Rounding

(2024, 01, 31) +_m 1 → (2024, 02, 31)

Rounding to valid dates required!

rounding mode \( r \ ::= \uparrow | \downarrow | \perp \)

expressions \( e ::= v | e +_\delta n | \text{rnd}_r e \)

\[
\begin{align*}
\text{rnd}_\uparrow (2024, 02, 31) &= (2024, 03, 01) \\
\text{rnd}_\downarrow (2024, 02, 31) &= (2024, 02, 29) \\
\text{rnd}_\perp (2024, 02, 31) &= \perp
\end{align*}
\]

Coreutils-like rounding not defined here
Semantics – Rounding

**Round-Noop**

\[ 1 \leq d \leq \text{nb\_days}(y, m) \]

\[
\text{rnd}_r(y, m, d) \rightarrow (y, m, d)
\]
Semantics – Rounding

**ROUND-NOOP**

\[ 1 \leq d \leq \text{nb}_\text{days}(y, m) \]

\[ \text{rnd}_r(y, m, d) \rightarrow (y, m, d) \]

**ROUND-DOWN**

\[ d > \text{nb}_\text{days}(y, m) \]

\[ \text{rnd}_\downarrow(y, m, d) \rightarrow (y, m, \text{nb}_\text{days}(y, m)) \]
Semantics – Rounding

**ROUND-NOOP**

\[ 1 \leq d \leq \text{nb}_\text{days}(y, m) \]

\[ \text{rnd}_r(y, m, d) \rightarrow (y, m, d) \]

**ROUND-DOWN**

\[ d > \text{nb}_\text{days}(y, m) \]

\[ \text{rnd}_\downarrow(y, m, d) \rightarrow (y, m, \text{nb}_\text{days}(y, m)) \]

**ROUND-UP**

\[ d > \text{nb}_\text{days}(y, m) \]

\[ (y, m, d) + m \ 1 \rightarrow^* (y', m', d') \]

\[ \text{rnd}_\uparrow(y, m, d) \rightarrow (y', m', 1) \]
Semantics – Rounding

**ROUND-NOOP**

\[
1 \leq d \leq \text{nb\_days}(y, m)
\]

\[
\text{rnd}_r(y, m, d) \rightarrow (y, m, d)
\]

**ROUND-DOWN**

\[
d > \text{nb\_days}(y, m)
\]

\[
\text{rnd}_\downarrow(y, m, d) \rightarrow (y, m, \text{nb\_days}(y, m))
\]

**ROUND-UP**

\[
d > \text{nb\_days}(y, m)
\]

\[
(y, m, d) + m \ 1 \rightarrow^* (y', m', d')
\]

\[
\text{rnd}_\uparrow(y, m, d) \rightarrow (y', m', 1)
\]

**ROUND-ERR2**

\[
d > \text{nb\_days}(y, m)
\]

\[
\text{rnd}_\bot(y, m, d) \rightarrow \bot
\]
Date-period addition

Given a period \((ys, ms, ds)\):

\[
e + _r (ys, ms, ds) ::= \text{rnd}_r((e + _y ys) + _m ms) + _d ds
\]
## Date-period addition

Given a period \((ys, ms, ds)\):

\[
e +_r (ys, ms, ds) ::= \text{rnd}_r((e +_y ys) +_m ms) +_d ds
\]

Avoids double rounding
Semantics

Date-period addition

Given a period \((ys, ms, ds)\):

\[ e \, +_{r} \, (ys, ms, ds) \, ::= \, \text{rnd}_{r}( (e +_{y} ys) +_{m} ms) +_{d} ds \]

Avoids double rounding

Ambiguous expression

A date expression \(e\) is ambiguous iff \(\text{rnd}_{\bot}(e) \, \overset{\star}{\rightarrow} \, \bot\)
Date-period addition

Given a period \((ys, ms, ds)\):

\[
e +_r (ys, ms, ds) ::= \text{rnd}_r((e +_y ys) +_m ms) +_d ds
\]

Avoids double rounding

Ambiguous expression

A date expression \(e\) is ambiguous iff \(\text{rnd}_\bot(e) \not\rightarrow \bot\)

iff roundings \(e\) yield different values
Formalized Properties
Non-properties

Commutativity of addition

\[(2024, 03, 31) \uparrow 1m \uparrow 1d = (2024, 05, 01) \uparrow 1d = (2024, 05, 02)\]
Non-properties

Commutativity of addition

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

\[
(2024, 03, 31) +_{\uparrow} 1d +_{\uparrow} 1m = (2024, 04, 01) +_{\uparrow} 1m = (2024, 05, 01)
\]
Non-properties

### Commutativity of addition

\[(2024, 03, 31) \uparrow 1m \uparrow 1d = (2024, 05, 01) \uparrow 1d = (2024, 05, 02)\]

\[(2024, 03, 31) \uparrow 1d \uparrow 1m = (2024, 04, 01) \uparrow 1m = (2024, 05, 01)\]

### “Associativity” of addition

\[(2024, 03, 31) \uparrow 1m \uparrow 1m = (2024, 05, 01) \uparrow 1m = (2024, 06, 01)\]
## Non-properties

### Commutativity of addition

\[
\begin{align*}
(2024, 03, 31) +^1 m +^1 d &= (2024, 05, 01) +^1 d = (2024, 05, 02) \\
(2024, 03, 31) +^1 d +^1 m &= (2024, 04, 01) +^1 m = (2024, 05, 01)
\end{align*}
\]

### “Associativity” of addition

\[
\begin{align*}
(2024, 03, 31) +^1 m +^1 m &= (2024, 05, 01) +^1 m = (2024, 06, 01) \\
(2024, 03, 31) + r 2 m &= (2024, 05, 31)
\end{align*}
\]
Formalized properties

All formalized with the F* proof assistant. More in the paper & artefact.
During our study, we used QCheck to test our intuition.
All formalized with the F* proof assistant. More in the paper & artefact.

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:

$$\text{valid}(d) \land d +^r p \rightarrow v \Rightarrow \text{valid}(v)$$
Formalized properties

All formalized with the F* proof assistant. More in the paper & artefact.

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:

\[
\text{valid}(d) \land d \oplus_r p \Rightarrow v \Rightarrow \text{valid}(v)
\]

Date addition is monotonic
For any dates \(d_1, d_2\), period \(p\), \(r \in \{\downarrow, \uparrow\}\), if \(d_1 < d_2\), then \(d_1 \oplus_r p \leq d_2 \oplus_r p\)
Formalized properties

All formalized with the F* proof assistant. More in the paper & artefact.
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:

$$valid(d) \land d + r \ p \overset{*}{\to} v \Rightarrow valid(v)$$

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

Loose bound in conclusion of monotonicity
$$(2024, 03, 30) + \downarrow 1m = (2024, 04, 30) = (2024, 03, 31) + \downarrow 1m$$
### Rounding is monotonic

For all date $d$, period $p$:

1. $d + \downarrow p \leq d + \uparrow p$

2. $d + \perp p \neq \perp \Rightarrow d + \downarrow p = d + \uparrow p = d + \perp p$
Rounding is monotonic
For all date $d$, period $p$:

1. $d + \downarrow p \leq d + \uparrow p$
2. $d + \perp p \neq \perp \Rightarrow d + \downarrow p = d + \uparrow p = d + \perp p$

Equivalence of year and month addition
For all date $d$, for all integer $n$, $d +_y n = d +_m (12 \times n)$. 
Formalized properties (III)

Ambiguous month addition

For all valid date $d$, integer $n$ such that $d + m n \rightarrow (y, m, day)$:

$$\text{nb\_days}(y, m) < day \Leftrightarrow \text{rnd}_\bot((y, m, day)) \rightarrow \bot$$
Ambiguous month addition

For all valid date $d$, integer $n$ such that $d + m n \rightarrow (y, m, day)$:

$$nb\_days(y, m) < day \iff \text{rnd}_\perp((y, m, day)) \rightarrow \perp$$

Month addition is ambiguous iff

the resulting day exceeds the number of days of the resulting month
Ambiguous month addition

For all valid date $d$, integer $n$ such that $d + m \cdot n \rightarrow (y, m, day)$:

\[ \text{nb\_days}(y, m) < day \Leftrightarrow \text{rnd}\perp((y, m, day)) \rightarrow \perp \]

Month addition is ambiguous iff

the resulting day exceeds the number of days of the resulting month

\[ \Rightarrow \text{core result needed for our static analysis} \]
Rounding-insensitivity Static Analysis
### Meaningful ambiguities

When rounding up or down doesn’t change a computation

<table>
<thead>
<tr>
<th>d + 1 month</th>
<th>&lt;= April 15 2024</th>
</tr>
</thead>
</table>

Rounding choice can change comparisons

\[ d = \text{March 31 2024} \Rightarrow \text{Prove rounding-insensitivity of an expression} \]
<table>
<thead>
<tr>
<th>Meaningful ambiguities</th>
</tr>
</thead>
<tbody>
<tr>
<td>When rounding up or down doesn’t change a computation</td>
</tr>
<tr>
<td>[ d + 1 \text{ month} \leq \text{April 15 2024} ]</td>
</tr>
<tr>
<td>No rounding? Safe</td>
</tr>
</tbody>
</table>
When rounding up or down doesn’t change a computation

\[ d + 1 \text{ month} \leq \text{April 15 2024} \]

- No rounding? Safe
- Otherwise, the rounding of \( d + 1 \text{ month} \) will not change the comparison.
Meaningful ambiguities

When rounding up or down doesn’t change a computation

\[ d + 1 \text{ month} \leq \text{April 15 2024} \]

- No rounding? Safe
- Otherwise, the rounding of \( d + 1 \text{ month} \) will not change the comparison.

Rounding choice can change comparisons

\[ d + 1 \text{ month} \leq \text{April 30 2024} \]
Meaningful ambiguities

When rounding up or down doesn’t change a computation

\[ d + 1 \text{ month} \leq \text{April 15 2024} \]

- No rounding? Safe
- Otherwise, the rounding of \( d + 1 \text{ month} \) will not change the comparison.

Rounding choice can change comparisons

\[ d + 1 \text{ month} \leq \text{April 30 2024} \]

- Rounding-sensitive comparison \( d = \text{March 31 2024} \)
## Meaningful ambiguities

### When rounding up or down doesn’t change a computation

<table>
<thead>
<tr>
<th>Expression</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d + 1 \text{ month}$</td>
<td>$\leq \text{ April 15 2024}$</td>
</tr>
</tbody>
</table>

- No rounding? Safe
- Otherwise, the rounding of $d + 1 \text{ month}$ will not change the comparison.

### Rounding choice can change comparisons

<table>
<thead>
<tr>
<th>Expression</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d + 1 \text{ month}$</td>
<td>$\leq \text{ April 30 2024}$</td>
</tr>
</tbody>
</table>

- Rounding-sensitive comparison $d = \text{ March 31 2024}$

$\Rightarrow$ Prove rounding-insensitivity of an expression $e$,
### Meaningful ambiguities

#### When rounding up or down doesn’t change a computation

<table>
<thead>
<tr>
<th>d + 1 month &lt;= April 15 2024</th>
</tr>
</thead>
<tbody>
<tr>
<td>► No rounding? Safe</td>
</tr>
<tr>
<td>► Otherwise, the rounding of ( d + 1 \text{ month} ) will not change the comparison.</td>
</tr>
</tbody>
</table>

#### Rounding choice can change comparisons

<table>
<thead>
<tr>
<th>d + 1 month &lt;= April 30 2024</th>
</tr>
</thead>
<tbody>
<tr>
<td>► Rounding-sensitive comparison ( d = \text{March 31 2024} )</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \text{Prove rounding-insensitivity of an expression } e, \, \mathbb{E}_{\uparrow}[e] = \mathbb{E}_{\downarrow}[e] \]
Meaningful ambiguities

**When rounding up or down doesn’t change a computation**

\[ d + 1 \text{ month} \leq \text{April 15 2024} \]

- No rounding? Safe
- Otherwise, the rounding of \( d + 1 \text{ month} \) will not change the comparison.

**Rounding choice can change comparisons**

\[ d + 1 \text{ month} \leq \text{April 30 2024} \]

- Rounding-sensitive comparison \( d = \text{March 31 2024} \)

\[ \Rightarrow \text{Prove rounding-insensitivity of an expression } e, \mathbb{E}_\uparrow[e] = \mathbb{E}_\downarrow[e] \]

To reduce the need for costly legal interpretations
Rounding-insensitivity Static Analysis

Abstracting dates in a fixed rounding mode
YMD domain

- Defines addition, accessors, projection, lexicographic comparison
YMD domain

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

- Defines addition, accessors, projection, lexicographic comparison
- Translates constraints on dates into numerical constraints
date $d_1 \leadsto$ ghost numerical variables $d(d_1), m(d_1), y(d_1)$
YMD domain

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

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

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

Given a rounding mode, compute resulting dates from \( d\# +_m n \), where \( d\# \) represents a set of dates.

Soundly derived from the ambiguous addition theorem.
YMD domain – month addition

<table>
<thead>
<tr>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given a rounding mode, compute resulting dates from ( d^# + _m ^# n ), where ( d^# ) represents a set of dates.</td>
</tr>
</tbody>
</table>

Soundly derived from the ambiguous addition theorem.

Algorithm: compute resulting month, year, then 4 cases:

- No rounding,
- Rounding, 30-day month,
- Rounding, non-leap years 28 Feb,
- Rounding, leap years, 29 Feb.

Partitioning used in practice.
YMD domain – month addition (II)

```ocaml
let add_months (r: rnd) ((d, m, y): var) (nb_m: int) (abs: state): cases =
    let res_m: expr = 1 + (m - 1 + nb_m) % 12 in
    let res_y: expr = y + (m - 1 + nb_m) / 12 in
    switch abs abs
        [ d > 30 && is_one_of res_m [Apr; Jun; Sep; Nov],
          round r 30 res_m res_y;
        d > 28 && res_m = Feb && not (is_leap res_y),
          round r 28 res_m res_y;
        d > 29 && res_m = Feb && is_leap res_y,
          round r 29 res_m res_y;
        mk_true,
          mk_date d res_m res_y
        ]
```

Choosing the right numerical abstract domains

date d1 = rand_date(); date d2 = d1 + 1 month; rounding down.
Choosing the right numerical abstract domains

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

► No concrete values on d1
► Intervals would be imprecise

⇒ relational abstract domains needed!
Choosing the right numerical abstract domains

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

▶ No concrete values on d1
▶ Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:
Choosing the right numerical abstract domains

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

- No concrete values on d1
- Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:

- 30-day month

  \[ d(d1) = 31, \ m(d1) \in \{ \text{Mar, May, Aug, Oct} \}, \ m(d2) = m(d1) + 1, y(d2) = y(d1) \]
Choosing the right numerical abstract domains

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

- No concrete values on d1
- Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:

- 30-day month

  \[ d(d1) = 31, \ m(d1) \in \{ \text{Mar, May, Aug, Oct} \}, \ m(d2) = m(d1) + 1, y(d2) = y(d1) \]

  Bounded set of ints
Choosing the right numerical abstract domains

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

► No concrete values on d1
► Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:

► 30-day month

\[
d(d1) = 31, \ m(d1) \in \{\text{Mar, May, Aug, Oct}\}, \quad m(d2) = m(d1) + 1, \ y(d2) = y(d1)
\]

Bounded set of ints Polyhedra
Choosing the right numerical abstract domains

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

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

⇒ relational abstract domains needed!

4 cases apply, including:

- 30-day month

  \[ d(d1) = 31, \ m(d1) \in \{ \text{Mar, May, Aug, Oct} \}, \ m(d2) = m(d1) + 1, \ y(d2) = y(d1) \]

  Bounded set of ints

  Polyhedra

- No rounding \( d(d1) = d(d2), \ m(d2) \equiv_{12} m(d1) + 1, \ y(d1) \leq y(d2) \leq y(d1) + 1 \)
Choosing the right numerical abstract domains

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

- No concrete values on \(d1\)
- Intervals would be imprecise

\[\Rightarrow\] relational abstract domains needed!

4 cases apply, including:

- 30-day month

\[
d(d1) = 31, \quad m(d1) \in \{\text{Mar, May, Aug, Oct}\}, \quad m(d2) = m(d1) + 1, \quad y(d2) = y(d1)
\]

Bounded set of ints  Polyhedra

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

Linear congruence domain
Choosing the right numerical abstract domains

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

- No concrete values on \( d1 \)
- Intervals would be imprecise

\( \implies \) relational abstract domains needed!

4 cases apply, including:

- 30-day month

\[
d(d1) = 31, \ m(d1) \in \{\text{Mar}, \text{May}, \text{Aug}, \text{Oct}\}, \ m(d2) = m(d1) + 1, y(d2) = y(d1)
\]

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

\( ^1 \text{Actually,} \ 12 y(d1) + m(d1) \leq 12 y(d2) + 11 \land 12 y(d2) \leq 12 y(d1) + m(d1) + 1 \)
Rounding-insensitivity Static Analysis

Lifting to both rounding modes
Semantics on product programs with both rounding modes.

Semantics on product programs with both rounding modes.

\[ \mathbb{E}_r[e] : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\text{Val}), r \in \{\uparrow, \downarrow\} \]

Semantics on product programs with both rounding modes.

\[ \mathbb{E}_r[e] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\text{Val}), \ r \in \{\uparrow, \downarrow\} \sim \mathbb{E}_\downarrow[e] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\text{Val}^2) \]
Semantics on product programs with both rounding modes.

\[ \mathbb{E}_r[e] : \mathcal{P}(E) \rightarrow \mathcal{P}(\text{Val}), r \in \{\uparrow, \downarrow\} \quad \leadsto \quad \mathbb{E}_\downarrow[e] : \mathcal{P}(E^2) \rightarrow \mathcal{P}(\text{Val}^2) \]

\[ \mathbb{E}_\downarrow[e_1 + e_2](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in D} \{ (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) = \mathbb{E}_\downarrow[e_1]\rho_\uparrow, \]

\[ (v_2^\uparrow, v_2^\downarrow) = \mathbb{E}_\downarrow[e_2]\rho_\downarrow \} \]
Semantics on product programs with both rounding modes.

\[
E_r[e] : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\text{Val}), r \in \{\uparrow, \downarrow\} \quad \leadsto \quad E_{\downarrow}[e] : \mathcal{P}(\mathcal{E}^2) \rightarrow \mathcal{P}(\text{Val}^2)
\]

\[
E_{\downarrow}[e_1 + e_2](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in D} \{ (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) | (v_1^\uparrow, v_1^\downarrow) = E_{\downarrow}[e_1]_{\rho_\uparrow}, (v_2^\uparrow, v_2^\downarrow) = E_{\downarrow}[e_2]_{\rho_\downarrow} \}
\]

\[
E_{\downarrow}[\text{rand\_date()}](D) = \{ (d, d) | d \in \mathbb{Z}^3, \text{valid}(d) \}
\]
Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[e] : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\text{Val}), r \in \{\uparrow, \downarrow\} \sim \mathbb{E}_f[e] : \mathcal{P}(\mathcal{E}^2) \rightarrow \mathcal{P}(\text{Val}^2)$$

$$\mathbb{E}_f[e_1 + e_2](D) = \bigcup_{(\rho^\uparrow, \rho^\downarrow) \in \mathcal{D}} \{ (v^\uparrow_1 + v^\uparrow_2, v^\downarrow_1 + v^\downarrow_2) \mid (v^\uparrow_1, v^\downarrow_1) = \mathbb{E}_f[e_1] \rho^\uparrow, (v^\uparrow_2, v^\downarrow_2) = \mathbb{E}_f[e_2] \rho^\downarrow \}$$

$$\mathbb{E}_f[\text{rand_date()}](D) = \{ (d, d) \mid d \in \mathbb{Z}^3, \text{valid}(d) \}$$

sync(e) holds iff e is rounding-insensitive.
Semantics on product programs with both rounding modes.

\[ \mathbb{E}_r[e] : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\mathcal{Val}), r \in \{\uparrow, \downarrow\} \sim \mathbb{E}_↓[e] : \mathcal{P}(\mathcal{E}^2) \rightarrow \mathcal{P}(\mathcal{Val}^2) \]

\[ \mathbb{E}_↓[e_1 + e_2](D) = \bigcup_{(\rho_↑, \rho_↓) \in \mathcal{D}} \{ (v_1↑ + v_2↑, v_1↓ + v_2↓) \mid (v_1↑, v_1↓) = \mathbb{E}_↓[e_1] \rho_↑, (v_2↑, v_2↓) = \mathbb{E}_↓[e_2] \rho_↓ \} \]

\[ \mathbb{E}_↓[\text{rand\_date()}](D) = \{ (d, d) \mid d \in \mathbb{Z}^3, \text{valid}(d) \} \]

**sync(e)** holds iff \( e \) is rounding-insensitive.

\[ \mathbb{E}_↓[\text{sync}(e)](D) = \bigcup_{(\rho_↑, \rho_↓) \in \mathcal{D}} \{ (b_u == b_d, b_u == b_d) \mid (b_u, b_d) = \mathbb{E}_↓[e](\rho_↑, \rho_↓) \} \]

Semantics on product programs with both rounding modes.

\[
\mathbb{E}_r[e] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{V} \downarrow), r \in \{\uparrow, \downarrow\} \sim \mathbb{E}_\downarrow[e] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathcal{V}^2) \\
\mathbb{E}_\downarrow[e_1 + e_2](D) = \bigcup\limits_{(\rho_\uparrow, \rho_\downarrow) \in D} \{ (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) = \mathbb{E}_\downarrow[e_1]\rho_\uparrow, (v_2^\uparrow, v_2^\downarrow) = \mathbb{E}_\downarrow[e_2]\rho_\downarrow \}
\]

\[
\mathbb{E}_\downarrow[\text{rand_date()}](D) = \{ (d, d) \mid d \in \mathbb{Z}^3, \text{valid}(d) \}
\]

\begin{itemize}
  \item \textbf{sync}(e) holds iff } e \text{ is rounding-insensitive.}
  \[
  \mathbb{E}_\downarrow[\text{sync}(e)](D) = \bigcup\limits_{(\rho_\uparrow, \rho_\downarrow) \in D} \{ (b_u = b_d, b_u = b_d) \mid (b_u, b_d) = \mathbb{E}_\downarrow[e](\rho_\uparrow, \rho_\downarrow) \}
  \]
\end{itemize}

Abstract double semantics

Shallow variable duplication depending on their rounding mode.
Abstract double semantics

Shallow variable duplication depending on their rounding mode.

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

Shallow variable duplication depending on their rounding mode.

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

- No rounding

\[ d(d1) = d(d2) \quad m(d2) \equiv_{12} m(d1) + 1 \quad y(d1) \leq y(d2) \leq y(d1) + 1 \]
Abstract double semantics

Shallow variable duplication depending on their rounding mode.

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

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

- 30-day month
  \[ d(d1) = 31, \quad m(d1) \in \{ \text{Mar, May, Aug, Sep} \} \]
  \[ \downarrow d(d2) = 30, \quad \downarrow m(d2) \in \{ \text{Apr, Jun, Sep, Nov} \}, \quad \downarrow m(d2) = m(d1) + 1 \]
  \[ \uparrow d(d2) = 1, \quad \uparrow m(d2) \in \{ \text{May, Jul, Oct, Dec} \}, \quad \uparrow m(d2) = m(d1) + 2 \]
  \[ \downarrow y(d2) = \uparrow y(d2) = y(d1) \]
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
Implementation into Mopsa

- Open-source static analysis platform
- C, Python, C+Python programs
- [gitlab.com/mopsa/mopsa-analyzer](gitlab.com/mopsa/mopsa-analyzer)
Implementation into Mopsa

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

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

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

```python
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));
```

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
```python
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));
```

Computed, actual counter-example

Desynchronization detected:

$\uparrow$ month(limit) = 3, $\uparrow$ day(limit) = 1,
$\downarrow$ month(limit) = 2, $\downarrow$ day(limit) = 1,
$\uparrow$ month(intermediate) = 3, $\uparrow$ day(intermediate) = 1,
$\downarrow$ month(intermediate) = 2, $\downarrow$ day(intermediate) = 28,
year(birthday) = [4] 0, month(birthday) = 2, day(birthday) = 29,
$\uparrow$ year(intermediate) = $\uparrow$ year(limit) = $\uparrow$ year(birthday) + 2,
$\downarrow$ year(intermediate) = $\downarrow$ year(limit) = $\downarrow$ year(birthday) + 2.
```
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));
```

Desynchronization detected:
- month(limit) = 3, day(limit) = 1
- month(intermediate) = 2, day(intermediate) = 28
- month(birthday) = 2, day(birthday) = 29
- year(birthday) = 0
- year(current) = year(intermediate)
- year(limit) = year(birthday) + 2

Computed, actual counter-example:
- current is in Feb. of year $y$
- birthday is 29 Feb. of leap year $y$
- intermediate is either 28 Feb. or 1 March of $y$
- limit is either 1 Feb. or 1 March of $y$
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));

Desynchronization detected:

- month(limit) = 3, day(limit) = 1
- month(intermediate) = 2, day(intermediate) = 28
- month(birthday) = 2, day(birthday) = 29
- year(birthday) = [4] 0, year(intermediate) = 2
- year(current) = year(intermediate) = year(limit) = year(birthday) + 2

Computed, actual counter-example:

- current is in Feb. of year y
- birthday is 29 Feb. of leap year y − 2
```python
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));
```

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
def rand_date():
    pass

def first_day_of(date):
    pass

def sync(current, limit):
    return current < limit

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));

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

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
Case Study: French Housing Benefits
Contributions to Catala

- Date-rounding library `dates-calc`
Contributions to Catala

- Date-rounding library `dates-calc`
- Scope-level rounding mode configuration
Case Study – Catala for the French Housing Benefits

Contributions to Catala

- Date-rounding library `dates-calc`
- Scope-level rounding mode configuration
- Connection with static analysis
## Case Study – Catala for the French Housing Benefits

### Contributions to Catala

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

### French Housing Benefits

20,000 Loc of Catala code (including text spec.)
Date ambiguity detection pipeline

file.catala → Slicing → date-sensitive computations → Prog. gen. → progs.u → Mopsa

• 2 rounding-sensitive cases detected
• Intra-scope extraction for now
• Manual inter-scope extraction
• 16 additional cases:
  ▶ 10 can be proved safe (assuming current_date ≥ 2023)
  ▶ Other are real issues

⚠️ + Hints
Date ambiguity detection pipeline

2 rounding-sensitive cases detected
Date ambiguity detection pipeline

2 rounding-sensitive cases detected

Intra-scope extraction for now
Date ambiguity detection pipeline

**2 rounding-sensitive cases detected**

**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
Conclusion
Related Work

### Survey of implementations

- Java, **boost** round down
- Python **stdlib**: no month addition
- Inconsistency in spreadsheets

Recent Rocq formalization: Ana, Bedmar, Rodríguez, Reyes, Buñuel, and Joosten. "UTC Time, Formally Verified". CPP 2024
Related Work

Survey of implementations

- Java, **boost** round down
- Python **stdlib**: no month addition
- Inconsistency in spreadsheets

Timezones, leap seconds & co.

Recent Rocq formalization: Ana, Bedmar, Rodríguez, Reyes, Buñuel, and Joosten. “UTC Time, Formally Verified”. CPP 2024
## Related Work

### Survey of implementations
- Java, `boost` round down
- Python `stdlib`: no month addition
- Inconsistency in spreadsheets

### Floating-point arithmetic
- FP widely used & more complex!
- Different rounding modes
- No analysis of rounding-sensitivity?

### Timezones, leap seconds & co.
Recent Rocq formalization: Ana, Bedmar, Rodríguez, Reyes, Buñuel, and Joosten. “UTC Time, Formally Verified”. CPP 2024
Conclusion

- Formal semantics of date computations

[Artefact & paper available!](https://rmonat.fr/publication/24_esop/)

"Automatic Verification of Catala programs" (AVoCAT) project
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)

Artefact & paper available!
https://rmonat.fr/publication/24_esop/
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)
- Theorems verified in F*

Artefact & paper available! [https://rmonat.fr/publication/24_esop/]

"Automatic Verification of Catala programs" (AVoCAT) project
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)
- Theorems verified in F*
- Ambiguity-detection static analysis using Mopsa

Artefact & paper available! [https://rmonat.fr/publication/24_esop/](https://rmonat.fr/publication/24_esop/)

"Automatic Verification of Catala programs" (AVoCAT) project
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)
- Theorems verified in F⋆
- Ambiguity-detection static analysis using Mopsa
- Case study on Catala encoding of French housing benefits
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)
- Theorems verified in F*
- Ambiguity-detection static analysis using Mopsa
- Case study on Catala encoding of French housing benefits
- Comparison with mainstream implementations
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)
- Theorems verified in F⋆
- Ambiguity-detection static analysis using Mopsa
- Case study on Catala encoding of French housing benefits
- Comparison with mainstream implementations

Artefact & paper available! https://rmonat.fr/publication/24_esop/
Conclusion

- Formal semantics of date computations
- OCaml library implementing our semantics (also in Python now!)
- Theorems verified in F*
- Ambiguity-detection static analysis using Mopsa
- Case study on Catala encoding of French housing benefits
- Comparison with mainstream implementations

Artefact & paper available! https://rmonat.fr/publication/24_esop/

“Automatic Verification of Catala programs” (AVoCAT) project