# 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

# Legal implementations

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

### Catala

- a DSL for computational laws
- providing transparency
- easing maintenance
- through interdisciplinary work

# News: Catala is getting closer to the real world

# News: Catala is getting closer to the real world

## Partnerships with government agencies

## 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 ressources.

# 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 ressources.

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

# 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 ressources.

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

# 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 ressources.

**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

# 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 ressources.

**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

# 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 ressources.

**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)

# Handling Date Computations in Catala

# Computing dates

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

# 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
```

# Computing dates

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

# Computing dates

```
$ 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

# 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

$\Longrightarrow$ Formal, flexible semantics required!

# 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

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

# Outline

# Semantics

# Semantics – Values

| | | | |
|---|---|---|---|
| values | $v$ | $::=$ | $(y, m, d) \mid \perp$ |
| date unit | $\delta$ | $::=$ | $y \mid m \mid d$ |
| expressions | $e$ | $::=$ | $v \mid e \ +_{\delta} \ n$ |

# Semantics – Values

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

$$\text{nb\_days}(y, m) = \begin{cases} 29 \text{ if } m = 2 \land \text{is\_leap}(y) \\ 28 \text{ if } m = 2 \land \neg\text{is\_leap}(y) \\ 30 \text{ if } m \in \{\, \texttt{Apr}, \texttt{Jun}, \texttt{Sep}, \texttt{Nov} \,\} \\ 31 \text{ otherwise} \end{cases}$$

# Semantics – invalid dates

Day additions with invalid day number propagate errors

## Semantics – invalid dates

**Day additions with invalid day number propagate errors**

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

# Semantics – invalid dates

**Day additions with invalid day number propagate errors**

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

$$\frac{day > \text{nb\_days}(y, m)}{(y, m, day) +_d n \rightarrow \bot}\ \textsc{Add-Days-Err2}$$

# Semantics – some cases of month addition

$$\textsc{Add-Month}\quad \frac{1 \leq m + n \leq 12}{(y, m, d) +_m n \to (y, m + n, d)}$$

# Semantics – some cases of month addition

$$\frac{1 \leq m + n \leq 12}{(y, m, d) +_m n \to (y, m + n, d)}\ \textsc{Add-Month}$$

$$\frac{m + n > 12}{(y, m, d) +_m n \to (y + 1, m, d) +_m (n - 12)}\ \textsc{Add-Month-Over}$$

## Semantics – some cases of month addition

$$\frac{1 \leq m + n \leq 12}{(y, m, d) +_m n \rightarrow (y, m + n, d)}\ \textsc{Add-Month} \qquad \frac{m + n > 12}{(y, m, d) +_m n \rightarrow (y + 1, m, d) +_m (n - 12)}\ \textsc{Add-Month-Over}$$

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

# Semantics – Rounding

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

# Semantics – Rounding

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

Rounding to valid dates required!

# Semantics – Rounding

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

Rounding to valid dates required!

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

# Semantics – Rounding

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

Rounding to valid dates required!

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

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

# Semantics – Rounding

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

Rounding to valid dates required!

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

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

# Semantics – Rounding

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

Rounding to valid dates required!

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

$$\begin{aligned} \mathrm{rnd}_\uparrow(2024, 02, 31) &= (2024, 03, 01) \\ \mathrm{rnd}_\downarrow(2024, 02, 31) &= (2024, 02, 29) \\ \mathrm{rnd}_\perp(2024, 02, 31) &= \perp \end{aligned}$$

# Semantics – Rounding

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

Rounding to valid dates required!

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

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

Coreutils-like rounding not defined here

# Semantics – Rounding

$$\textsc{Round-Noop}\quad \frac{1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \to (y, m, d)}$$

# Semantics – Rounding

$$\textsc{Round-Noop}\ \frac{1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \rightarrow (y, m, d)}$$

$$\textsc{Round-Down}\ \frac{d > \text{nb\_days}(y, m)}{\text{rnd}_{\downarrow}(y, m, d) \rightarrow (y, m, \text{nb\_days}(y, m))}$$

# Semantics – Rounding

Round-Noop

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

Round-Down

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

Round-Up

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

$$\frac{\textsc{Round-Noop}\quad 1 \leq d \leq \text{nb\_days}(y, m)}{\text{rnd}_r(y, m, d) \rightarrow (y, m, d)}$$

$$\frac{\textsc{Round-Down}\quad d > \text{nb\_days}(y, m)}{\text{rnd}_\downarrow(y, m, d) \rightarrow (y, m, \text{nb\_days}(y, m))}$$

$$\frac{\textsc{Round-Up}\quad d > \text{nb\_days}(y, m) \qquad (y, m, d) +_m 1 \xrightarrow{*} (y', m', d')}{\text{rnd}_\uparrow(y, m, d) \rightarrow (y', m', 1)}$$

$$\frac{\textsc{Round-Err2}\quad d > \text{nb\_days}(y, m)}{\text{rnd}_\perp(y, m, d) \rightarrow \perp}$$

**Date-period addition**

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

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

# Semantics

**Date-period addition**

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

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

Avoids double rounding

# Semantics

## Date-period addition

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

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

Avoids double rounding

## Ambiguous expression

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

### Date-period addition

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

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

Avoids double rounding

### Ambiguous expression

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

# Formalized Properties

## Non-properties

### Commutativity of addition

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

# Non-properties

**Commutativity of addition**

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

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

# Non-properties

**Commutativity of addition**

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

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

**“Associativity” of addition**

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

## Non-properties

**Commutativity of addition**

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

**"Associativity" of addition**

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

# Formalized properties

**All formalized with the F$^\star$ proof assistant. More in the paper & artefact.**

During our study, we used QCheck to test our intuition.

**All formalized with the F$^\star$ 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:

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

## Formalized properties

**All formalized with the F$^\star$ 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:

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

**Date addition is monotonic**

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

# 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:

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

**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**

$(2024, 03, 30) +_\downarrow 1m = (2024, 04, 30) = (2024, 03, 31) +_\downarrow 1m$

# Formalized properties (II)

**Rounding is monotonic**

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

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

# Formalized properties (II)

**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 $n$, $d +_y n = d +_m (12 * n)$.

# Formalized properties (III)

## Ambiguous month addition

For all valid date $d$, integer $n$ such that $d +_m n \xrightarrow{*} (y, m, day)$:

$$\text{nb\_days}(y, m) < day \Leftrightarrow \text{rnd}_\perp((y, m, day)) \xrightarrow{*} \perp$$

# Formalized properties (III)

## Ambiguous month addition

For all valid date $d$, integer $n$ such that $d +_m n \xrightarrow{*} (y, m, day)$:

$$\text{nb\_days}(y, m) < day \Leftrightarrow \text{rnd}_\perp((y, m, day)) \xrightarrow{*} \perp$$

Month addition is ambiguous iff

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

# Formalized properties (III)

**Ambiguous month addition**

For all valid date $d$, integer $n$ such that $d +_m n \xrightarrow{*} (y, m, day)$:

$$\text{nb\_days}(y, m) < day \Leftrightarrow \text{rnd}_\perp((y, m, day)) \xrightarrow{*} \perp$$

Month addition is ambiguous iff

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

$\Longrightarrow$ core result needed for our static analysis

# Rounding-insensitivity Static Analysis

# Meaningful ambiguities

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

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

# Meaningful ambiguities

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

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

- No rounding? Safe

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

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

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

## Meaningful ambiguities

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

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

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

### Rounding choice can change comparisons

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

# Meaningful ambiguities

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

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

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

## Rounding choice can change comparisons

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

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

# Meaningful ambiguities

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

`d + 1 month <= April 15 2024`

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

## Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

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

# Meaningful ambiguities

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

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

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

## Rounding choice can change comparisons

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

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

$\Longrightarrow$ 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 month <= April 15 2024`

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

## Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

$\Longrightarrow$ 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 $d_1 \rightsquigarrow$ ghost numerical variables $\mathsf{d}(d_1), \mathsf{m}(d_1), \mathsf{y}(d_1)$

## YMD domain

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

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

## YMD domain – month addition

**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

**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.

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)

```
type case = expr * state
type cases = case list
let switch abs =
  List.map (fun (cond : expr, k : state -> case) -> k (assume cond abs))

let add_months (r: rnd) ((d, m, y): var^3) (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
  [
    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

$\Longrightarrow$ 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

$\implies$ 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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

# 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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

# 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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

# 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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

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

# 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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

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

## 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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

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

[1] Actually, $12\,\mathrm{y}(\mathtt{d1}) + \mathrm{m}(\mathtt{d1}) \leq 12\,\mathrm{y}(\mathtt{d2}) + 11 \wedge 12\,\mathrm{y}(\mathtt{d2}) \leq 12\,\mathrm{y}(\mathtt{d1}) + \mathrm{m}(\mathtt{d1}) + 1$

# Rounding-insensitivity Static Analysis

Lifting to both rounding modes

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

## Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathrm{Val}), r \in \{\uparrow, \downarrow\} \quad \rightsquigarrow$$

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathsf{Val}), r \in \{\uparrow, \downarrow\} \quad \rightsquigarrow \quad \mathbb{E}_{\updownarrow}[\![e]\!] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathsf{Val}^2)$$

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathsf{Val}), r \in \{\uparrow, \downarrow\} \quad \rightsquigarrow \quad \mathbb{E}_{\updownarrow}[\![e]\!] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathsf{Val}^2)$$

$$\begin{aligned}\mathbb{E}_{\updownarrow}[\![e_1 + e_2]\!](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in \mathcal{D}} \{\, (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) = \mathbb{E}_{\updownarrow}[\![e_1]\!]\rho_\uparrow,& \\ (v_2^\uparrow, v_2^\downarrow) = \mathbb{E}_{\updownarrow}[\![e_2]\!]\rho_\downarrow \,\}&\end{aligned}$$

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathrm{Val}), r \in \{\uparrow, \downarrow\} \quad \rightsquigarrow \quad \mathbb{E}_{\updownarrow}[\![e]\!] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathrm{Val}^2)$$

$$\begin{aligned}\mathbb{E}_{\updownarrow}[\![e_1 + e_2]\!](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in \mathcal{D}} \{ (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) &= \mathbb{E}_{\updownarrow}[\![e_1]\!]\rho_\uparrow, \\ (v_2^\uparrow, v_2^\downarrow) &= \mathbb{E}_{\updownarrow}[\![e_2]\!]\rho_\downarrow \}\end{aligned}$$

$$\mathbb{E}_{\updownarrow}[\![\mathrm{rand\_date}()]\!](D) = \{ (d, d) \mid d \in \mathbb{Z}^3, \mathit{valid}(d) \}$$

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathrm{Val}), r \in \{\uparrow, \downarrow\} \quad \leadsto \quad \mathbb{E}_\updownarrow[\![e]\!] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathrm{Val}^2)$$

$$\begin{aligned}\mathbb{E}_\updownarrow[\![e_1 + e_2]\!](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in \mathcal{D}} \{\, (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) = \mathbb{E}_\updownarrow[\![e_1]\!]\rho_\uparrow, \\ (v_2^\uparrow, v_2^\downarrow) = \mathbb{E}_\updownarrow[\![e_2]\!]\rho_\downarrow \,\}\end{aligned}$$

$$\mathbb{E}_\updownarrow[\![\mathrm{rand\_date}()]\!](D) = \{\, (d, d) \mid d \in \mathbb{Z}^3, \mathit{valid}(d) \,\}$$

- $\mathtt{sync}(e)$ holds iff $e$ is rounding-insensitive.

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathsf{Val}), r \in \{\uparrow, \downarrow\} \quad \rightsquigarrow \quad \mathbb{E}_{\updownarrow}[\![e]\!] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathsf{Val}^2)$$

$$\begin{aligned}\mathbb{E}_{\updownarrow}[\![e_1 + e_2]\!](D) = \bigcup_{(\rho_\uparrow,\rho_\downarrow)\in\mathcal{D}} \{\, (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) = \mathbb{E}_{\updownarrow}[\![e_1]\!]\rho_\uparrow, \\ (v_2^\uparrow, v_2^\downarrow) = \mathbb{E}_{\updownarrow}[\![e_2]\!]\rho_\downarrow \,\}\end{aligned}$$

$$\mathbb{E}_{\updownarrow}[\![\mathsf{rand\_date()}]\!](D) = \{\, (d, d) \mid d \in \mathbb{Z}^3, \mathit{valid}(d) \,\}$$

- $\mathtt{sync}(e)$ holds iff $e$ is rounding-insensitive.

$$\mathbb{E}_{\updownarrow}[\![\mathsf{sync}(e)]\!](D) = \bigcup_{(\rho_\uparrow,\rho_\downarrow)\in\mathcal{D}} \{\, (b_u == b_d, b_u == b_d) \mid (b_u, b_d) = \mathbb{E}_{\updownarrow}[\![e]\!](\rho_\uparrow, \rho_\downarrow) \,\}$$

# Back to rounding-insensitivity detection

- Semantics on product programs with both rounding modes.

$$\mathbb{E}_r[\![e]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathrm{Val}), r \in \{\uparrow, \downarrow\} \quad \rightsquigarrow \quad \mathbb{E}_{\updownarrow}[\![e]\!] : \mathcal{P}(\mathcal{E}^2) \to \mathcal{P}(\mathrm{Val}^2)$$

$$\begin{aligned}\mathbb{E}_{\updownarrow}[\![e_1 + e_2]\!](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in \mathcal{D}} \{\, (v_1^\uparrow + v_2^\uparrow, v_1^\downarrow + v_2^\downarrow) \mid (v_1^\uparrow, v_1^\downarrow) = \mathbb{E}_{\updownarrow}[\![e_1]\!]\rho_\uparrow, \\ (v_2^\uparrow, v_2^\downarrow) = \mathbb{E}_{\updownarrow}[\![e_2]\!]\rho_\downarrow \,\}\end{aligned}$$

$$\mathbb{E}_{\updownarrow}[\![\mathrm{rand\_date()}]\!](D) = \{\, (d, d) \mid d \in \mathbb{Z}^3, \mathit{valid}(d) \,\}$$

- $\mathtt{sync}(e)$ holds iff $e$ is rounding-insensitive.

$$\mathbb{E}_{\updownarrow}[\![\mathrm{sync}(e)]\!](D) = \bigcup_{(\rho_\uparrow, \rho_\downarrow) \in \mathcal{D}} \{\, (b_u == b_d, b_u == b_d) \mid (b_u, b_d) = \mathbb{E}_{\updownarrow}[\![e]\!](\rho_\uparrow, \rho_\downarrow) \,\}$$

- Inspired by Delmas, Ouadjaout, and Miné. “Static Analysis of Endian Portability by Abstract Interpretation”. SAS 2021.

# 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

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

## Abstract double semantics

Shallow variable duplication depending on their rounding mode.

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

- No rounding

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

- 30-day month

$$\mathrm{d}(d1) = 31, \mathrm{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Sep}\}$$

$$\downarrow\mathrm{d}(d2) = 30, \downarrow\mathrm{m}(d2) \in \{\mathtt{Apr}, \mathtt{Jun}, \mathtt{Sep}, \mathtt{Nov}\}, \downarrow\mathrm{m}(d2) = \mathrm{m}(d1) + 1$$

$$\uparrow\mathrm{d}(d2) = 1, \uparrow\mathrm{m}(d2) \in \{\mathtt{May}, \mathtt{Jul}, \mathtt{Oct}, \mathtt{Dec}\}, \uparrow\mathrm{m}(d2) = \mathrm{m}(d1) + 2$$

$$\downarrow\mathrm{y}(d2) = \uparrow\mathrm{y}(d2) = \mathrm{y}(d1)$$

# Implementation into Mopsa

- Open-source static analysis platform

# Implementation into Mopsa

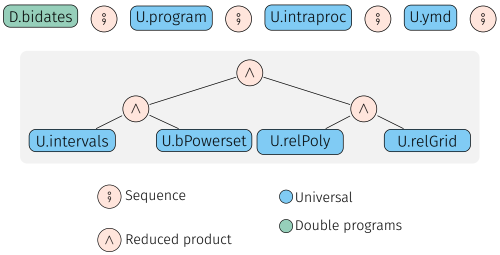
- Open-source static analysis platform
- C, Python, C+Python programs

# Implementation into Mopsa

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

# Implementation into Mopsa

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



# Extracted sample from French housing benefits

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

# Extracted sample from French housing benefits

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

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

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

# Extracted sample from French housing benefits

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

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

Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,                                        ,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) = ↓year(limit) = year(birthday) + 2
```

**Computed, actual counter-example**

```
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));
```

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

Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,                                          ,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) = ↓year(limit) = year(birthday) + 2
```

## Computed, actual counter-example

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

# Extracted sample from French housing benefits

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

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

Desynchronization detecte
↑month(limit) = 3, ↑day(l
↑month(intermediate) = 3,                                   ,
↓day(intermediate) = 28,
year(birthday) =[4] 0, mo
year(current) = ↑year(int
= ↓year(intermediate) = ↓year(limit) = year(birthday) + 2
```

## Computed, actual counter-example

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

## Extracted sample from French housing benefits

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

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

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

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

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

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

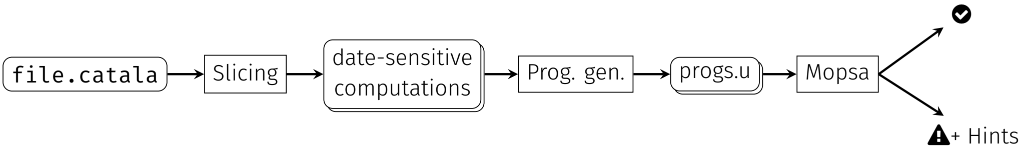
## Contributions to Catala

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

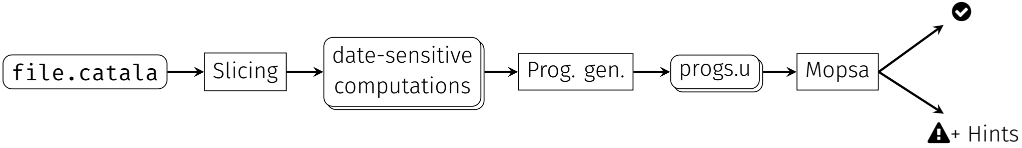
## French Housing Benefits

20,000 Loc of Catala code (including text spec.)

# Date ambiguity detection pipeline

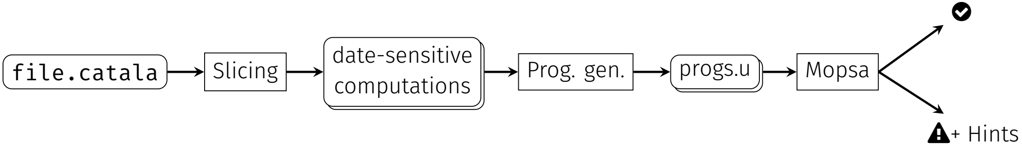


# Date ambiguity detection pipeline




2 rounding-sensitive cases detected

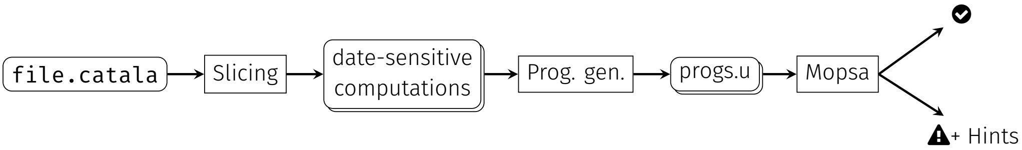
# Date ambiguity detection pipeline




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

# 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

# Conclusion

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

# Conclusion

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

# Conclusion

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

# 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/`

“**A**utomatic **V**erification **o**f **Cat**ala programs” (AVoCAT) project