Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law

Raphaël Monat, Aymeric Fromherz, Denis Merigoux

`rmonat.fr`

ESOP
11 April 2024

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
- easing maintenance
- providing transparency

Legal implementations

Some legal implementations are critical software: taxes, benefits

Catala

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

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

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

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

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

Semantics – invalid day number

Day additions with invalid day number propagate errors

Semantics – invalid day number

Day additions with invalid day number propagate errors

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

Semantics – invalid day number

Day additions with invalid day number propagate errors

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

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

Semantics – some cases of month addition

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

Semantics – some cases of month addition

Add-Month

$$\frac{1 \leq mo + n \leq 12}{(y, mo, d) +_m n \rightarrow (y, mo + n, d)}$$

Add-Month-Over

$$\frac{mo + n > 12}{(y, mo, d) +_m n \rightarrow (y + 1, mo, d) +_m (n - 12)}$$

Semantics – some cases of month addition

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

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

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!

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

Semantics – Rounding

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

Rounding to valid dates required!

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

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

Semantics – Rounding

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

Rounding to valid dates required!

rounding mode	r	$::=$	$\uparrow \mid \downarrow \mid \bot$
expressions	e	$::=$	$v \mid e +_\delta n \mid \mathrm{rnd}_r\, e$

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

Semantics – Rounding

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

Rounding to valid dates required!

rounding mode	r	$::=$	$\uparrow \mid \downarrow \mid \bot$
expressions	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

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

Rounding to valid dates required!

rounding mode	r	$::=$	$\uparrow \mid \downarrow \mid \bot$
expressions	e	$::=$	$v \mid e +_\delta n \mid \mathrm{rnd}_r\, e$

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

Coreutils-like rounding not defined here

Semantics – Rounding

Round-Noop

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

Semantics – Rounding

Round-Noop

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

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

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

Round-Down

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

Round-Up

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

Semantics – Rounding

Round-Noop

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

Round-Down

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

Round-Up

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

Round-Err2

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

Semantics

Date-period addition

Given a period (ys, ms, ds):

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

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 $\mathrm{rnd}_\bot(e) \xrightarrow{*} \bot$

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

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

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

"Associativity" of addition

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

$$(2024, 03, 31) +_{r} 2\text{m} = (2024, 05, 31)$$

Formalized properties

All formalized with the F* proof assistant.

- More in the paper & artefact.

Formalized properties

All formalized with the F* proof assistant.

- More in the paper & artefact.
- During our study, we used QCheck to test our intuition.

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:

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

Rounding-insensitivity Static Analysis

Meaningful ambiguities

Rounding choice can change comparisons

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

Meaningful ambiguities

Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

Meaningful ambiguities

Rounding choice can change comparisons

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

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

When rounding up or down doesn’t change a computation

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

Meaningful ambiguities

Rounding choice can change comparisons

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

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

When rounding up or down doesn't change a computation

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

- No rounding? Safe

Meaningful ambiguities

Rounding choice can change comparisons

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

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

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

Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

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.

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

Meaningful ambiguities

Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

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.

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

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$

Meaningful ambiguities

Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

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.

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

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$ encoded as $\mathtt{sync}(e)$

Meaningful ambiguities

Rounding choice can change comparisons

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

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

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.

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

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$ encoded as `sync`(e)
- Considering product programs with both rounding modes

Delmas, Ouadjaout, and Miné. "Static Analysis of Endian Portability by Abstract Interpretation". SAS 2021

Meaningful ambiguities

Rounding choice can change comparisons

`d + 1 month <= April 30 2024`

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

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.

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

- $\mathbb{E}_{\uparrow}[\![e]\!] = \mathbb{E}_{\downarrow}[\![e]\!]$ encoded as $\mathtt{sync}(e)$
- Considering product programs with both rounding modes
- Will reduce the need for costly legal interpretations

Delmas, Ouadjaout, and Miné. "Static Analysis of Endian Portability by Abstract Interpretation". SAS 2021

- Defines addition, accessors, projection, lexicographic comparison

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

- Defines addition, accessors, projection, lexicographic comparison
- Translates constraints on dates into numerical constraints
  date $d_1 \rightsquigarrow$ ghost numerical variables $\mathsf{d}(d_1), \mathsf{m}(d_1), \mathsf{y}(d_1)$

- 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

- 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

Transfer function computing $(d, m, y) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
```

Transfer function computing $(d, m, y) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define exprs corresponding to the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
```

Transfer function computing $(d, m, y) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define exprs corresponding to the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d > 30 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;






  ]
```

Transfer function computing (d, m, y) $+^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define exprs corresponding to the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d > 30 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*) (*^^^^^^^^^^^^^^*)
    (*********** guard condition *************) (* continuation *)




  ]
```

Transfer function computing $(d, m, y) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define exprs corresponding to the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d > 30 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*) (*^^^^^^^^^^^^^^*)
    (*********** guard condition *************) (* continuation *)
    (* ~> continuation (assume guard) *)




  ]
```

Transfer function computing $(\mathsf{d}, \mathsf{m}, \mathsf{y}) +^{\#}$ nb_m in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define exprs corresponding to the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d > 30 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (* Case 2: round resulting date to 28/02/Y, Y is not leap *)
    d > 28 && r_m = Feb && not (is_leap r_y), round 28 r_m r_y;
    (* Case 3: round resulting date to 29/02/Y, Y is leap *)
    d > 29 && r_m = Feb && is_leap r_y, round 29 r_m r_y;


  ]
```

Transfer function computing $(\mathsf{d}, \mathsf{m}, \mathsf{y}) +^{\#} \mathsf{nb\_m}$ in abstract state abs

```
let add_months ((d, m, y): var^3) (nb_m: int) (abs: state) =
  (* Define exprs corresponding to the resulting month, year *)
  let r_m : expr = 1 + (m - 1 + nb_m) % 12 in
  let r_y : expr = y + (m - 1 + nb_m) / 12 in
  (* Abstract switch with four different (guard, continuation) *)
  switch abs [
    (* Case 1: round resulting date in 30-day month *)
    d > 30 && is_one_of r_m [Apr;Jun;Sep;Nov], round 30 r_m r_y;
    (* Case 2: round resulting date to 28/02/Y, Y is not leap *)
    d > 28 && r_m = Feb && not (is_leap r_y), round 28 r_m r_y;
    (* Case 3: round resulting date to 29/02/Y, Y is leap *)
    d > 29 && r_m = Feb && is_leap r_y, round 29 r_m r_y;
    (* Case 4: no rounding *)
    mk_true, mk_date d r_m r_y;
  ]
```

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

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

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

$\Longrightarrow$ relational abstract domains needed!

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

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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

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

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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathsf{d}(d1) = 31,\ \mathsf{m}(d1) \in \{\texttt{Mar}, \texttt{May}, \texttt{Aug}, \texttt{Oct}\}, \mathsf{d}(d2) = 30,\ \mathsf{m}(d2) = \mathsf{m}(d1) + 1, \mathsf{y}(d2) = \mathsf{y}(d1)$$

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

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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathrm{d}(d1) = 31,\ \underbrace{\mathrm{m}(d1) \in \{\,\texttt{Mar}, \texttt{May}, \texttt{Aug}, \texttt{Oct}\,\}}_{\text{Bounded set of ints}},\ \mathrm{d}(d2) = 30,\ \mathrm{m}(d2) = \mathrm{m}(d1) + 1, \mathrm{y}(d2) = \mathrm{y}(d1)$$

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

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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathrm{d}(d1) = 31,\ \underbrace{\mathrm{m}(d1) \in \{\texttt{Mar}, \texttt{May}, \texttt{Aug}, \texttt{Oct}\}}_{\text{Bounded set of ints}},\ \mathrm{d}(d2) = 30,\ \underbrace{\mathrm{m}(d2) = \mathrm{m}(d1) + 1, \mathrm{y}(d2) = \mathrm{y}(d1)}_{\text{Polyhedra}}$$

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

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

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathsf{d}(d1) = 31,\ \underbrace{\mathsf{m}(d1) \in \{\mathtt{Mar}, \mathtt{May}, \mathtt{Aug}, \mathtt{Oct}\}}_{\text{Bounded set of ints}}, \mathsf{d}(d2) = 30,\ \underbrace{\mathsf{m}(d2) = \mathsf{m}(d1) + 1, \mathsf{y}(d2) = \mathsf{y}(d1)}_{\text{Polyhedra}}$$

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

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

- No concrete constraints on **d1**
- Intervals would be imprecise

$\Longrightarrow$ relational abstract domains needed!

4 cases apply, including:

- 30-day month

$$\mathrm{d}(d1) = 31, \underbrace{\mathrm{m}(d1) \in \{\texttt{Mar}, \texttt{May}, \texttt{Aug}, \texttt{Oct}\}}_{\text{Bounded set of ints}}, \mathrm{d}(d2) = 30, \underbrace{\mathrm{m}(d2) = \mathrm{m}(d1) + 1, \mathrm{y}(d2) = \mathrm{y}(d1)}_{\text{Polyhedra}}$$

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

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

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

Abstract double semantics

Moving to double programs

- Analyze the program in both rounding modes
- 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

Moving to double programs

- Analyze the program in both rounding modes
- Shallow variable duplication depending on their rounding mode

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

- No rounding

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

- 30-day month

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

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

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

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

Implementation into Mopsa

- Open-source static analysis platform

Implementation into Mopsa

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

Implementation into Mopsa

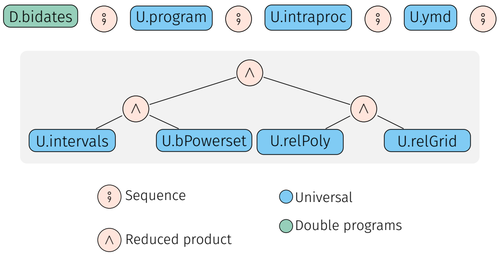
- Open-source static analysis platform
- C, Python, C+Python programs
- `gitlab.com/mopsa/mopsa-analyzer`

Implementation into Mopsa

- Open-source static analysis platform
- C, Python, C+Python programs
- `gitlab.com/mopsa/mopsa-analyzer`
- Winner of *SoftwareSystems* @ SV-Comp'24

Implementation into Mopsa

- Open-source static analysis platform
- C, Python, C+Python programs
- `gitlab.com/mopsa/mopsa-analyzer`
- Winner of *SoftwareSystems* @ SV-Comp'24



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));
```

```
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) = ↓
```

Computed, actual counter-example

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));
```

```
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

```
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 < limit));
                  ^^^^^^^^^^
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

```
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, m
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*

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
- `limit` is either 1 Feb. or 1 March of y

Case Study: French Housing Benefits

Catala, a DSL for computational laws

Article D823-20 of the French building regulations

The moving allowance is awarded to individuals or households with at least three children born or to be born and who move into a new home entitled to one of the personal housing allowances during a period between the first day of the calendar month following the third month of pregnancy for a child of rank three or more and the last day of the month preceding that in which the child reaches his or her second birthday.
This allowance is payable if the right to assistance is acquired within six months of the date of moving in.

````
```catala
scope MovingAllowanceEligibility:
 definition condition_moving_period under condition
  (match form.birthdate_third_child_or_more with pattern
  -- MoreThan3Children of date_of_birth_or_pregnancy:
   (match date_of_birth_or_pregnancy with pattern
    -- DateOfBirth of birthday
     current_date < (first_day_of_month of (birthday + 2 year))
    # ...
   )
  )
consequence fulfilled
```
````


Merigoux, Chataing, and Protzenko. “Catala: a programming language for the law”. 2021
Merigoux. “Experience report: implementing a real-world, medium-sized program derived from a legislative specification”. 2023

Catala, a DSL for computational laws

Article D823-20 of the French building regulations

The moving allowance is awarded to individuals or households with at least three children born or to be born and who move into a new home entitled to one of the personal housing allowances during a period between the first day of the calendar month following the third month of pregnancy for a child of rank three or more and the last day of the month preceding that in which the child reaches his or her second birthday.
This allowance is payable if the right to assistance is acquired within six months of the date of moving in.

````
```catala
scope MovingAllowanceEligibility:
 definition condition_moving_period under condition
  (match form.birthdate_third_child_or_more with pattern
  -- MoreThan3Children of date_of_birth_or_pregnancy:
   (match date_of_birth_or_pregnancy with pattern
    -- DateOfBirth of birthday
     current_date < (first_day_of_month of (birthday + 2 year))
    # ...
   )
  )
consequence fulfilled
```
````

- Literate programming
- Lawyer-developer duos
- Default logic tailored to the law
- Housing benefits: 20kLoC (incl. law)

Merigoux, Chataing, and Protzenko. "Catala: a programming language for the law". 2021
Merigoux. "Experience report: implementing a real-world, medium-sized program derived from a legislative specification". 2023

Contributions to Catala

- Date-rounding library `dates-calc`

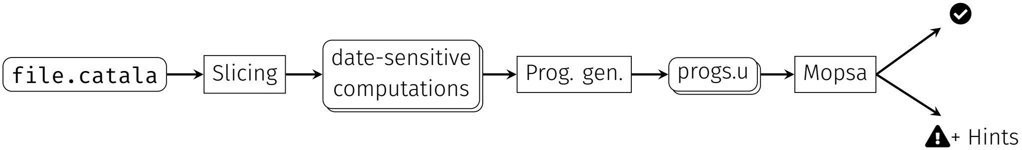
Contributions to Catala

- Date-rounding library `dates-calc`
- Scope-level rounding mode configuration

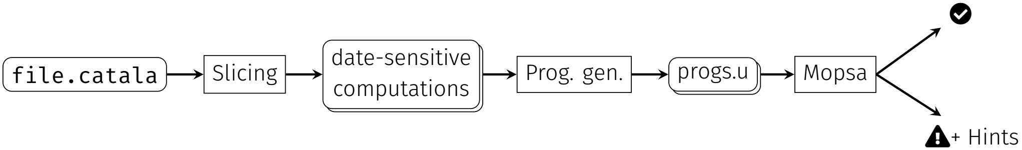
Contributions to Catala

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

Date ambiguity detection pipeline

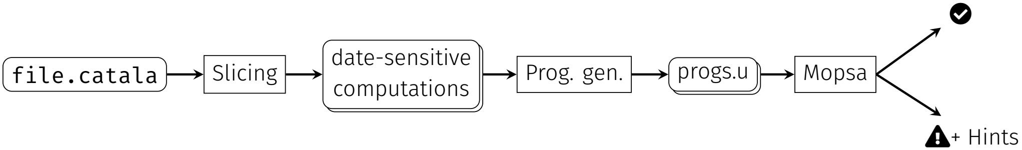


Date ambiguity detection pipeline



2 rounding-sensitive cases detected

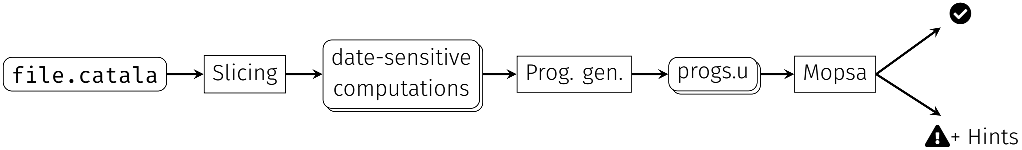
Date ambiguity detection pipeline



2 rounding-sensitive cases detected

No false alarms

Date ambiguity detection pipeline

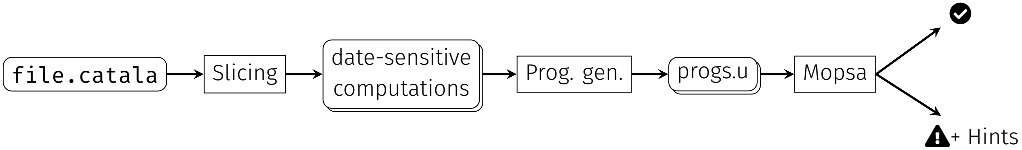


2 rounding-sensitive cases detected

No false alarms

Intra-scope extraction for now

Date ambiguity detection pipeline



2 rounding-sensitive cases detected

No false alarms

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

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*

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

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

Paper & artefact available! `rmonat.fr/esop24/`

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

Paper & artefact available! `rmonat.fr/esop24/`

“**A**utomatic **V**erification **o**f **Cat**ala programs” (AVoCat) project funded by Inria