Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law

Raphaël Monat, Aymeric Fromherz, Denis Merigoux

rmonat.fr
Some legal implementations are critical software: taxes, benefits
Some legal implementations are **critical** software: taxes, benefits

Catala

- a DSL for computational laws
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
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?!
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
### 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

⇒ 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

⇒ 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 \( v ::= (y, m, d) | \perp \)

date unit \( \delta ::= y | m | d \)

expressions \( e ::= v | e +_\delta n \)

\[
\text{nb\_days}(y, m) = \begin{cases} 
29 & \text{if } m = 2 \land \text{is\_leap}(y) \\
28 & \text{if } m = 2 \land \neg\text{is\_leap}(y) \\
30 & \text{if } m \in \{ \text{Apr}, \text{Jun}, \text{Sep}, \text{Nov} \} \\
31 & \text{otherwise}
\end{cases}
\]
Semantics – invalid day number

Day additions with invalid day number propagate errors
Semantics – invalid day number

Day additions with invalid day number propagate errors

\[ \text{ADD-DAYS-Err1} \]

\[
\text{day} < 1
\]

\[ (y, m, \text{day}) +_d n \rightarrow \bot \]
Day additions with invalid day number propagate errors

\[ \text{ADD-DAYS-ERR1} \]
\[ \text{day} < 1 \quad (y, m, \text{day}) +_{d} n \rightarrow \bot \]

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

ADD-MONTH

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

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

\[
\text{Add-Month} \quad \frac{1 \leq mo + n \leq 12}{(y, mo, d) +_m n \rightarrow (y, mo + n, d)}
\]

\[
\text{Add-Month-Over} \quad \frac{mo + n > 12}{(y, mo, d) +_m n \rightarrow (y + 1, mo, d) +_m (n - 12)}
\]
Semantics – some cases of month addition

**Add-Month**

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

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

**Add-Month-Over**

\[mo + n > 12\]

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

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 → (2024, 02, 31)

Rounding to valid dates required!
Semantics – Rounding

(2024, 01, 31) \(+_{m} 1 \rightarrow (2024, 02, 31)\)

Rounding to valid dates required!

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

expressions \( e ::= v | e +_{\delta} n | \text{rnd}_{r} e \)
Semantics – Rounding

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

Rounding to valid dates required!

\[
\text{rounding mode } \quad r ::= \uparrow | \downarrow | \bot
\]

\[
\text{expressions } \quad e ::= v | e +_\delta n | \text{rnd}_r e
\]

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

(2024, 01, 31) +_m 1 → (2024, 02, 31)

Rounding to valid dates required!

rounding mode  \( r \) ::= ↑ | ↓ | ⊥

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

\[ \text{rnd}_↑(2024, 02, 31) = (2024, 03, 01) \]
\[ \text{rnd}_↓(2024, 02, 31) = (2024, 02, 29) \]
Semantics – Rounding

\[(2024, 01, 31) +_{m} 1 \rightarrow (2024, 02, 31)\]

Rounding to valid dates required!

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

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

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

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

rounding mode   \( r \) ::= ↑ | ↓ | ⊥
expressions      \( e \) ::= v | e +δ n | rnd_r e

\[
\begin{align*}
\text{rnd}_↑(2024, 02, 31) &= (2024, 03, 01) \\
\text{rnd}_↓(2024, 02, 31) &= (2024, 02, 29) \\
\text{rnd}_⊥(2024, 02, 31) &= ⊥
\end{align*}
\]

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

\textbf{ROUND-NOOP}  
\[ 1 \leq d \leq \text{nb}_\text{days}(y, m) \]  
\[ \text{rnd}_r(y, m, d) \to (y, m, d) \]

\textbf{ROUND-DOWN}  
\[ d > \text{nb}_\text{days}(y, m) \]  
\[ \text{rnd}_\downarrow(y, m, d) \to (y, m, \text{nb}_\text{days}(y, m)) \]
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**

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

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

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

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

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

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

**ROUND-ERR2**

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

\[ \text{rnd}_\bot(y, m, d) \rightarrow \bot \]
Semantics

Date-period addition

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

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

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

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

Avoids double rounding

### Ambiguous expression

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

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

Commutativity of addition

(2024, 03, 31) +↑ 1m +↑ 1d = (2024, 05, 01) +↑ 1d = (2024, 05, 02)
### Non-properties

#### Commutativity of addition

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(2024, 03, 31) +_1 m +_1 d$</td>
<td>$(2024, 05, 01) +_1 d = (2024, 05, 02)$</td>
</tr>
<tr>
<td>$(2024, 03, 31) +_1 d +_1 m$</td>
<td>$(2024, 04, 01) +_1 m = (2024, 05, 01)$</td>
</tr>
</tbody>
</table>
### Non-properties

<table>
<thead>
<tr>
<th><strong>Commutativity of addition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>((2024, 03, 31) + \uparrow 1m + \uparrow 1d = (2024, 05, 01) + \uparrow 1d = (2024, 05, 02))</td>
</tr>
<tr>
<td>((2024, 03, 31) + \uparrow 1d + \uparrow 1m = (2024, 04, 01) + \uparrow 1m = (2024, 05, 01))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>“Associativity” of addition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>((2024, 03, 31) + \uparrow 1m + \uparrow 1m = (2024, 05, 01) + \uparrow 1m = (2024, 06, 01))</td>
</tr>
</tbody>
</table>
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

\[(2024, 03, 31) \uparrow 1m + \uparrow 1m = (2024, 05, 01) + \uparrow 1m = (2024, 06, 01)\]

\[(2024, 03, 31) + r 2m = (2024, 05, 31)\]
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:

\[
\text{valid}(d) \land d + rp \rightarrow v \implies \text{valid}(v)
\]
Formalized properties

All formalized with the F* proof assistant.

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

$$\text{valid}(d) \land d +_r p \xrightarrow{*} v \Rightarrow \text{valid}(v)$$
Rounding-insensitivity Static Analysis
Rounding choice can change comparisons

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

Rounding choice can change comparisons

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

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

When rounding up or down doesn't change a computation

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

- No rounding? Safe

- Otherwise, the rounding of \( d + 1 \text{ month} \) will not change the comparison.

Prove rounding-insensitivity of an expression \( e \)

- \( E \uparrow_J e K = E \downarrow_J e K \)

Encoded as \( \text{sync}(e) \)

Considering product programs with both rounding modes

Will reduce the need for costly legal interpretations
### Meaningful ambiguities

#### Rounding choice can change comparisons

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d + 1 \text{ month} \leq \text{April 30 2024}$</td>
<td>Rounding-sensitive comparison $d = \text{March 31 2024}$</td>
</tr>
</tbody>
</table>

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

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

---

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

Rounding choice can change comparisons

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

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

When rounding up or down doesn’t change a computation

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

- No rounding? Safe

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

Rounding choice can change comparisons

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

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

When rounding up or down doesn’t change a computation

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

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

Rounding choice can change comparisons

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

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

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.

\[ \implies \text{Prove rounding-insensitivity of an expression } e \]
Meaningful ambiguities

Rounding choice can change comparisons

\[ d + 1 \text{ month} \leq \text{April 30 2024} \]
- Rounding-sensitive comparison \( d = \text{March 31 2024} \)

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.

\[ \implies \text{Prove rounding-insensitivity of an expression } e \]
- \[ E_{\uparrow}[e] = E_{\downarrow}[e] \]
Meaningful ambiguities

Rounding choice can change comparisons

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

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

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.

\[ \Rightarrow \text{Prove rounding-insensitivity of an expression } e \]

- \( E_{\uparrow}[e] = E_{\downarrow}[e] \) encoded as \( \text{sync}(e) \)
Meaningful ambiguities

### Rounding choice can change comparisons

\[ d + 1 \text{ month} \leq \text{April 30 2024} \]
- Rounding-sensitive comparison \( d = \text{March 31 2024} \)

### 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.

\[ \Rightarrow \text{Prove rounding-insensitivity} \text{ of an expression } e \]
- \( E_\uparrow[e] = E_\downarrow[e] \) encoded as \( \text{sync}(e) \)
- 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

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

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

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.

⇒ Prove rounding-insensitivity of an expression \( e \)

▶ \( E_{\uparrow}[e] = E_{\downarrow}[e] \) encoded as \texttt{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
YMD domain

Fixed rounding mode

Defines addition, accessors, projection, lexicographic comparison

Translates constraints on dates into numerical constraints
d \( \rightarrow \) ghost numerical variables
d, m, y

Acts as a functor lifting a numerical abstract domain
d \( \in [1, 31] \wedge m \in [1, 12] \wedge y = 2024 \): all valid dates of 2024
YMD domain

- Defines addition, accessors, projection, lexicographic comparison

Fixed rounding mode
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 $d(d_1), m(d_1), y(d_1)$
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 $d_1 \rightsquigarrow$ ghost numerical variables $d(d_1), m(d_1), y(d_1)$
- Acts as a functor lifting a numerical abstract domain

\[
d(d_1) \in [1, 31] \land m(d_1) \in [1, 12] \land y(d_1) = 2024: \text{ all valid dates of 2024}
\]
Transfer function computing \((d, m, y) + \# \text{nb}_m\) in abstract state \(\text{abs}\)

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

YMD domain – month addition

Fixed round mode

Partitioning used in practice.

Soundly derived from the ambiguous addition theorem.
Transfer function computing \((d, m, y) + \# \text{nb\_m}\) in abstract state \text{abs}

```ocaml
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
```

Partitioning used in practice.

Soundly derived from the ambiguous addition theorem.
Transfer function computing \((d, m, y) \oplus \#\ nb_m\) in abstract state abs

```ocaml
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) \mod 12 in
  let r_y : expr = y + (m - 1 + nb_m) \div 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 \((d, m, y) + \#\ n_b_m\) in abstract state \(abs\)

```plaintext
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 *************)
    (*********** guard condition *************)
  ]
```
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 \((d, m, y) + \# \text{nb}_m\) in abstract state \text{abs}

```lean
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 &\& \text{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 &\& \text{r}_m = \text{Feb} &\& \text{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 &\& \text{r}_m = \text{Feb} &\& \text{is_leap r}_y, round 29 \ r_m\ r_y;
]
```

Partitioning used in practice.

Soundly derived from the ambiguous addition theorem.
Transfer function computing \((d, m, y) +\# \ nb\_m\) in abstract state \(abs\)

```plaintext
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;
]
```

Partitioning used in practice.
Soundly derived from the ambiguous addition theorem.
Choosing the right numerical abstract domains

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

No concrete constraints on $d_1$ ⇒ Intervals would be imprecise ⇒ relational abstract domains needed!

4 cases apply, including:

- 30-day month $d(d_1) = 31, m(d_1) \in \{\text{Mar}, \text{May}, \text{Aug}, \text{Oct}\}$

- Bounded set of ints, $d(d_2) = 30, m(d_2) = m(d_1) + 1, y(d_2) = y(d_1)$

- No rounding $d(d_1) = d(d_2), m(d_2) \equiv 12 m(d_1) + 1$

- Linear congruence domain, $y(d_1) \leq y(d_2) \leq y(d_1) + 1$
Choosing the right numerical abstract domains

Fixed round mode

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

- No concrete constraints on d1
- Intervals would be imprecise

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

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

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

\[\rightarrow\text{ relational abstract domains needed!}\]

4 cases apply, including:

- 30-day month

\[d(d1) = 31, \ m(d1) \in \{\text{Mar, May, Aug, Oct}\}, d(d2) = 30, \ m(d2) = m(d1) + 1, y(d2) = y(d1)\]
Choosing the right numerical abstract domains

**Fixed round mode**

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

- No concrete constraints on \( d_1 \)
- Intervals would be imprecise

\[ \Rightarrow \text{relational abstract domains needed!} \]

4 cases apply, including:

- 30-day month

\[
\begin{align*}
d(d1) &= 31, \ m(d1) \in \{\text{Mar, May, Aug, Oct}\}, \\
d(d2) &= 30, \ m(d2) = m(d1) + 1, \ y(d2) = y(d1)
\end{align*}
\]

Bounded set of ints
Choosing the right numerical abstract domains

Fixed round mode

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

- No concrete constraints on d1
- Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:

- 30-day month

\[
d(d1) = 31, \quad m(d1) \in \{\text{Mar, May, Aug, Oct}\}, \quad d(d2) = 30, \quad m(d2) = m(d1) + 1, \quad y(d2) = y(d1)
\]

- Bounded set of ints
- Polyhedra
Choosing the right numerical abstract domains

Date \(d_1 = \text{rand\_date}()\); date \(d_2 = d_1 + 1 \text{ month}\); rounding down.

- No concrete constraints on \(d_1\)
- Intervals would be imprecise

\[ \implies \text{relational abstract domains needed!} \]

4 cases apply, including:

- 30-day month

\[
d(d_1) = 31, \ m(d_1) \in \{\text{Mar, May, Aug, Oct}\}, d(d_2) = 30, \ m(d_2) = m(d_1) + 1, y(d_2) = y(d_1)
\]

- No rounding  \(d(d_1) = d(d_2), \ m(d_2) \equiv_{12} m(d_1) + 1, y(d_1) \leq y(d_2) \leq y(d_1) + 1\)
Choosing the right numerical abstract domains

Fixed round mode

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

▶ No concrete constraints on d1
▶ Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:

▶ 30-day month

\[
d(d1) = 31, \quad m(d1) \in \{\text{Mar, May, Aug, Oct}\}, \quad d(d2) = 30, \quad m(d2) = m(d1) + 1, y(d2) = y(d1)
\]

Bounded set of ints

Polyhedra

▶ No rounding

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

Linear congruence domain
Abstract double semantics

Moving to double programs

► Analyze the program in both rounding modes

d1 = rand_date(); d2 = d1 + 1 month;

da(d1) = d(d2) m(d2) ≡ 12 m(d1) + 1 y(d1) ≤ y(d2) ≤ y(d1) + 1

d1 = 31, m(d1) ∈ {Mar, May, Aug, Sep}

↓ d2 = 30, ↓ m(d2) ∈ {Apr, Jun, Sep, Nov}, ↓ m(d2) = m(d1) + 1

↑ d2 = 1, ↑ m(d2) ∈ {May, Jul, Oct, Dec}, ↑ m(d2) = m(d1) + 2

↓ y(d2) = ↑ y(d2) = y(d1)
Abstract double semantics

Moving to double programs

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

```java
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

```java
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

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

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

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

▶ 30-day month

\[
d(d1) = 31, m(d1) \in \{ \text{Mar, May, Aug, Sep} \}
\]

\[
\downarrow d(d2) = 30, \downarrow m(d2) \in \{ \text{Apr, Jun, Sep, Nov} \}, \downarrow m(d2) = m(d1) + 1
\]

\[
\uparrow d(d2) = 1, \uparrow m(d2) \in \{ \text{May, Jul, Oct, Dec} \}, \uparrow m(d2) = m(d1) + 2
\]

\[
\downarrow y(d2) = \uparrow y(d2) = y(d1)
\]
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](https://gitlab.com/mopsa/mopsa-analyzer)
- Winner of *SoftwareSystems* @ SV-Comp’24
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));
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(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
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) = 3,
↓ day(intermediate) = 28,
year(birthday) = [4] 0, month(birthday) = 2,
year(current) = ↑ year(intermediate) = year(birthday) + 2
= ↓ year(intermediate) = ↓ year(limit) = year(birthday) + 2
```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,
- year(birthday) = 2019,
- 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

19
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
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
```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));
```

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
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. “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
```

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
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
Date ambiguity detection pipeline

2 rounding-sensitive cases detected
No false alarms
Intra-scope extraction for now
16 additional cases:
  ▶ 10 can be proved safe assuming current_date ≥ 2023
  ▶ Other are real issues

+ Hints
Date ambiguity detection pipeline

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

2 rounding-sensitive cases detected

No false alarms
Intra-scope extraction for now
16 additional cases:
▶ 10 can be proved safe assuming current_date ≥ 2023
▶ Other are real issues
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 \textit{current\_date} \geq 2023
- Other are real issues
Conclusion
Survey of implementations

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

Recent Rocq formalization: Ana, Bedmar, Rodrı́guez, 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
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

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
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!  ![AVoCat](rmonat.fr/esop24/)

“Automatic Verification of Catala programs” (AVoCat) project funded by Inria