Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law

Raphaël Monat, Aymeric Fromherz, Denis Merigoux

rmonat.fr

ESOP
11 April 2024
Some legal implementations are critical software: taxes, benefits
Legal implementations

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

<table>
<thead>
<tr>
<th>Catala</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ a DSL for computational laws</td>
</tr>
</tbody>
</table>
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
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
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

⇒ 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
Semantics – Values

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

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

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

values \( v \ ::= \ (y, m, d) | \bot \)
date unit \( \delta \ ::= \ y | m | d \)
expressions \( e \ ::= \ v | e +_\delta n \)

\[
\text{nb_days}(y, m) = \begin{cases} 
29 & \text{if } m = 2 \land \text{isLeap}(y) \\
28 & \text{if } m = 2 \land \neg \text{isLeap}(y) \\
30 & \text{if } m \in \{\text{April, June, September, November}\} \\
31 & \text{otherwise}
\end{cases}
\]
Semantics – invalid dates

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

$$\text{ADD-DAYS-ERR1}$$

\[
\frac{day < 1}{(y, m, day) +_d n \rightarrow \bot}
\]
Semantics – invalid dates

Day additions with invalid day number propagate errors

\[
\begin{align*}
\text{ADD-DAYS-ERR1} & : \quad day < 1 \\
(y, m, day) + d n & \rightarrow \bot
\end{align*}
\]

\[
\begin{align*}
\text{ADD-DAYS-ERR2} & : \quad day > \text{nb_days}(y, m) \\
(y, m, day) + d n & \rightarrow \bot
\end{align*}
\]
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

**Add-Month**

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

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

**Add-Month-Over**

\[ mo + n > 12 \]

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

\[
\text{Add-Month}
\begin{align*}
1 & \leq mo + n \leq 12 \\
(y, mo, d) +_m n & \rightarrow (y, mo + n, d)
\end{align*}
\]

\[
\text{Add-Month-Over}
\begin{align*}
mo + n & > 12 \\
(y, mo, d) +_m n & \rightarrow (y + 1, mo, d) +_m (n - 12)
\end{align*}
\]

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

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

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

Rounding to valid dates required!
Semantics – Rounding

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

Rounding to valid dates required!

rounding mode \( r \) ::= ↑ | ↓ | ⊥
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!

rounding mode \( r \ ::= \ ↑ | ↓ | \perp \)

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

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

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

rounding mode \quad r ::= \uparrow | \downarrow | \perp
expressions \quad e ::= v \mid e +_\delta n \mid \text{rnd}_r e

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

(2024, 01, 31) +_m 1 → (2024, 02, 31)

Rounding to valid dates required!

rounding mode \( r \) ::= ↑ | ↓ | ⊥

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

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

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

Rounding to valid dates required!

rounding mode \quad r ::= \uparrow | \downarrow | ⊥

expressions \quad e ::= v | e +δ n | \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) = ⊥

Coreutils-like rounding not defined here
Semantics – Rounding

\[
\text{ROUND-NOOP} \\
1 \leq d \leq \text{nb_days}(y, m) \\
\therefore \text{rnd}_r(y, m, d) \rightarrow (y, m, d)
\]
Semantics – Rounding

**Round-Noop**

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

**Round-Down**

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

**Round-Up**

\[
d > \text{nb}_\text{days}(y, m) \\
\text{round}_\uparrow(y, m, d) \rightarrow (y, m', d'
\]
Semantics – Rounding

**Round-Noop**

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

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

**Round-Down**

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

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

**Round-Up**

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

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

\[ \text{rnd}_\uparrow(y, m, d) \rightarrow (y', m', 1) \]
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
\[ d > \text{nb\_days}(y, m) \]
\[ \text{rnd}_\perp(y, m, d) \rightarrow \perp \]
Date-period addition

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

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

Date-period addition

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

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

Avoids double rounding
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 \(\text{rnd}_\perp(e) \not\rightarrow \perp\)
Semantics

Date-period addition

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

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

Avoids double rounding

Ambiguous expression

A date expression \(e\) is ambiguous iff \(\text{rnd}_\bot(e) \overset{*}{\rightarrow} \bot\)

iff roundings \(e\) yield different values
Formalized Properties
Commutativity of addition

$$(2024, 03, 31) +_{↑1m} +_{↑1d} = (2024, 05, 01) +_{↑1d} = (2024, 05, 02)$$
Non-properties

<table>
<thead>
<tr>
<th>Commutativity of addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>((2024, 03, 31) + ↑1m + ↑1d = (2024, 05, 01) + ↑1d = (2024, 05, 02))</td>
</tr>
<tr>
<td>((2024, 03, 31) + ↑1d + ↑1m = (2024, 04, 01) + ↑1m = (2024, 05, 01))</td>
</tr>
</tbody>
</table>
Non-properties

**Commutativity of addition**

\[
(2024, 03, 31) +_{1m} +_{1d} = (2024, 05, 01) +_{1d} = (2024, 05, 02) \\
(2024, 03, 31) +_{1d} +_{1m} = (2024, 04, 01) +_{1m} = (2024, 05, 01)
\]

**“Associativity” of addition**

\[
(2024, 03, 31) +_{1m} +_{1m} = (2024, 05, 01) +_{1m} = (2024, 06, 01)
\]
## Non-properties

### Commutativity of addition

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>((2024, 03, 31) +<em>{1m} +</em>{1d} = (2024, 05, 01) +_{1d} = (2024, 05, 02))</td>
<td>((2024, 05, 01) +_{1m} = (2024, 05, 01))</td>
</tr>
<tr>
<td>((2024, 03, 31) +<em>{1d} +</em>{1m} = (2024, 04, 01) +_{1m} = (2024, 05, 01))</td>
<td></td>
</tr>
</tbody>
</table>

### “Associativity” of addition

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>((2024, 03, 31) +<em>{1m} +</em>{1m} = (2024, 05, 01) +_{1m} = (2024, 06, 01))</td>
<td>((2024, 03, 31) +<em>{1m} +</em>{1m} = (2024, 05, 01))</td>
</tr>
<tr>
<td>((2024, 03, 31) +_{2m} = (2024, 05, 31))</td>
<td>((2024, 05, 31))</td>
</tr>
</tbody>
</table>
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.
Formalized properties

All formalized with the F* proof assistant.

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

Ambiguous month addition

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

$$\text{nb\_days}(y, m) < day \Leftrightarrow \text{rnd}\bot((y, m, day)) \rightarrow \bot$$
Formalized properties

All formalized with the F* proof assistant.

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

Ambiguous month addition

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

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

Month addition is ambiguous iff

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

All formalized with the F* proof assistant.

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

Ambiguous month addition

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

\[
\text{nb\_days}(y, m) < day \iff \text{rnd}\bot((y, m, day)) \to \bot
\]

Month addition is ambiguous iff

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

\( \implies \) core result needed for our static analysis
Rounding-insensitivity Static Analysis
### Meaningful ambiguities

Rounding choice can change comparisons

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

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.

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

Considering product programs with both rounding modes

\[
E^{\uparrow} J e^{\downarrow} K = E^{\uparrow} J (e^{\downarrow} K)
\]

Will reduce the need for costly legal interpretations
Rounding choice can change comparisons

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

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

\[ d + 1 \text{ month} \leqslant \text{April 15 2024} \]
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
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.
# Meaningful ambiguities

## Rounding choice can change comparisons

<table>
<thead>
<tr>
<th>d + 1 month &lt;= April 30 2024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounding-sensitive comparison d = March 31 2024</td>
</tr>
</tbody>
</table>

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

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

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.

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

<table>
<thead>
<tr>
<th>Expression</th>
<th>Date Comparison</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>Date Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d + 1 \text{ month} \leq \text{April 15 2024})</td>
<td>No rounding? Safe</td>
</tr>
<tr>
<td></td>
<td>Otherwise, the rounding of (d + \text{1 month}) will not change the comparison.</td>
</tr>
</tbody>
</table>

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

- Considering product programs with both rounding modes
- \(E_{\uparrow}[e] = E_{\downarrow}[e]\) encoded as \(\text{sync}(e)\)
Meaningful ambiguities

Rounding choice can change comparisons

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

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

When rounding up or down doesn’t change a computation

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

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

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

- Considering product programs with both rounding modes
- \( \mathbb{E}_{\uparrow}[e] = \mathbb{E}_{\downarrow}[e] \) encoded as \textit{sync}(e)
- 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(d_1) \rightarrow \text{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: all valid dates of 2024
YMD domain

▶ Defines addition, accessors, projection, lexicographic comparison

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

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

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

\( d(d_1) \in [1, 31] \land m(d_1) \in [1, 12] \land y(d_1) = 2024: \) all valid dates of 2024
Compute resulting dates of $d\# +_{m}^{n} n$, where $d\#$ represents a set of dates.

Soundly derived from the ambiguous addition theorem.
Compute resulting dates of $d^\# + m\# n$, where $d^\#$ represents a set of dates. Soundly derived from the ambiguous addition theorem.

Compute resulting month, year, then 4 cases:

- No rounding,
- Rounding, 30-day month,
- Rounding, non-leap years 28 Feb,
- Rounding, leap years, 29 Feb.
Compute resulting dates of $d\# +^m n$, where $d\#$ represents a set of dates.

Soundly derived from the ambiguous addition theorem.

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

- No concrete values on $d_1$
- Intervals would be imprecise $\Rightarrow$ relational abstract domains needed!

- 4 cases apply, including:
  - 30-day month: $d(d_1) = 31$, $m(d_1) \in \{\text{Mar, May, Aug, Oct}\}$
  - Bounded set of ints
  - $m(d_2) = m(d_1) + 1$, $y(d_2) = y(d_1)$
  - Polyhedra: $d(d_1) = d(d_2)$
  - Linear congruence domain: $y(d_1) \leq y(d_2) \leq y(d_1) + 1$
Choosing the right numerical abstract domains

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

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

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

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

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

⇒ relational abstract domains needed!

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

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

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

⇒ relational abstract domains needed!

4 cases apply, including:

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

Fixed round mode

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

- No concrete values on d1
- Intervals would be imprecise

⇒ relational abstract domains needed!

4 cases apply, including:

- 30-day month

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

\text{Bounded set of ints}
Choosing the right numerical abstract domains

<table>
<thead>
<tr>
<th>date d1 = rand_date(); date d2 = d1 + 1 month; rounding down.</th>
</tr>
</thead>
<tbody>
<tr>
<td>► No concrete values on d1</td>
</tr>
<tr>
<td>► Intervals would be imprecise</td>
</tr>
<tr>
<td>⇒ relational abstract domains needed!</td>
</tr>
</tbody>
</table>

4 cases apply, including:

► 30-day month

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

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

**Fixed round mode**

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

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

\[\implies\] relational abstract domains needed!

4 cases apply, including:

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

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

16
Choosing the right numerical abstract domains

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

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

⇒ relational abstract domains needed!

4 cases apply, including:

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

  Bounded set of ints


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

  Linear congruence domain

```
Abstract double semantics

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

\[d(d_1) = d(d_2) \equiv 12\]
\[m(d_1) \leq y(d_2) \leq y(d_1) + 1\]

\[d(d_1) = 31, m(d_1) \in \{\text{Mar}, \text{May}, \text{Aug}, \text{Sep}\}\]
\[d(d_2) = 30, m(d_2) \in \{\text{Apr}, \text{Jun}, \text{Sep}, \text{Nov}\}, \uparrow m(d_2) = m(d_1) + 1\]
\[↑ d(d_2) = 1, ↑ m(d_2) \in \{\text{May}, \text{Jul}, \text{Oct}, \text{Dec}\}, ↑ m(d_2) = m(d_1) + 2\]
\[↓ y(d_2) = ↑ y(d_2) = y(d_1)\]
Abstract double semantics

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

date $d_1 = \text{rand\_date}();$ date $d_2 = d_1 + 1 \text{ month};$ double semantics

- No rounding

$$d(d_1) = d(d_2) \quad m(d_2) \equiv_{12} m(d_1) + 1 \quad y(d_1) \leq y(d_2) \leq y(d_1) + 1$$
Abstract double semantics

\( \text{date } d_1 = \text{rand\_date()} \); \( \text{date } d_2 = d_1 + 1 \ \text{month}; \) double semantics

- No rounding

\[ d(d_1) = d(d_2) \quad m(d_2) \equiv_{12} m(d_1) + 1 \quad y(d_1) \leq y(d_2) \leq y(d_1) + 1 \]

- 30-day month

\[ d(d_1) = 31, m(d_1) \in \{ \text{Mar, May, Aug, Sep} \} \]
\[ \downarrow d(d_2) = 30, \downarrow m(d_2) \in \{ \text{Apr, Jun, Sep, Nov} \}, \downarrow m(d_2) = m(d_1) + 1 \]
\[ \uparrow d(d_2) = 1, \uparrow m(d_2) \in \{ \text{May, Jul, Oct, Dec} \}, \uparrow m(d_2) = m(d_1) + 2 \]
\[ \downarrow y(d_2) = \uparrow y(d_2) = y(d_1) \]
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));
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, month(current) = 2, day(current) = [1,29],
year(current) = ↑year(intermediate) = ↑year(limit)
= ↓year(intermediate) = ↓year(limit) = year(birthday) + 2
```python
date current = rand_date();
date birthday = rand_date();
date intermediate = birthday + [2 years, 0 months, 0 days];
date limit = first_day_of(intermediate);
assert(sync(current < limit));
```

Computed, actual counter-example

Desynchronization detected:

↑ month(limit) = 3, ↑ day(limit) = 1,
↑ month(intermediate) = 3, ↓ day(intermediate) = 28,
↓ month(birthday) = 2, day(birthday) = 29,
year(birthday) = [4] 0, month(current) = ↑ year(intermediate) = ▼
year(current) = ↑ year(intermediate) = ▼ year(limit) = year(birthday) + 2
```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 \( y \)
- intermediate is either 28 Feb. or 1 March of \( y \)
- limit is either 1 Feb. or 1 March of \( y \)

\( \uparrow \) month(limit) = 3, \( \uparrow \) day(limit) = 1, \( \downarrow \) month(birthday) = 2, day(birthday) = 29, year(birthday) = 0, month(current) = \( \uparrow \) 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));

Computed, actual counter-example

- **current** is in Feb. of year \( y \)
- **birthday** is 29 Feb. of leap year \( y - 2 \)

Desynchronization detected:

\[
\begin{align*}
\text{month}(\text{limit}) &= 3, \quad \text{day}(\text{limit}) = 1, \\
\text{month}(\text{intermediate}) &= 3, \quad \text{day}(\text{intermediate}) = 1, \\
\text{month}(\text{birthday}) &= 2, \quad \text{day}(\text{birthday}) = 29, \\
\text{year}(\text{birthday}) &= \text{year}(\text{intermediate}) = \text{year}(\text{limit}) = \text{year}(\text{intermediate}) = \text{year}(\text{limit}) = \text{year}(\text{birthday}) + 2
\end{align*}
\]
```python
date current = rand_date();
date birthday = rand_date();
date intermediate = birthday + [2 years, 0 months, 0 days];
date limit = first_day_of(intermediate);
assert(sync(current < limit));
```

Desynchronization detected: (current < limit).

Hints:
- ↑ month(limit) = 3, ↑ day(birthday) = 29
- ↑ month(intermediate) = 3, ↓ day(intermediate) = 28
- ↓ month(birthday) = 2, day(birthday) = 29
- ↑ year(intermediate) = year(birthday) + 2
- ↓ 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 \(y\)
```plaintext
1 date current = rand_date();
2 date birthday = rand_date();
3 date intermediate = birthday + [2 years, 0 months, 0 days];
4 date limit = first_day_of(intermediate);
5 assert(sync(current < limit));
```

Desynchronization detected:

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

Computed, actual counter-example:

- **current** is in Feb. of year y
- **birthday** is 29 Feb. of leap year y – 2
- **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
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

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

⚠️ + Hints
Date ambiguity detection pipeline

2 rounding-sensitive cases detected

No false alarms
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 \texttt{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

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, 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

- 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!
rmonat.fr/esop24/
Conclusion

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

Artefact & paper available!
rmonat.fr/esop24/

"Automatic Verification of Catala programs" project, funded by Inria
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

Artefact & paper 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

Artefact & paper available! rmonat.fr/esop24/

“Automatic Verification of Catala programs” project, funded by Inria
Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law Questions

Raphaël Monat, Aymeric Fromherz, Denis Merigoux
rmonat.fr