

Try-Mopsa

Relational Static Analysis in Your Pocket

Raphaël Monat
`rmonat.fr`



Modular Open Platform for Static Analysis [Jou+19]

`mopsa.lip6.fr` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)



Modular Open Platform for Static Analysis [Jou+19]

`mopsa.lip6.fr` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- ▶ Allows users to choose their composition of abstract domains



Modular Open Platform for Static Analysis [Jou+19]

`mopsa.lip6.fr` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- ▶ Allows users to choose their composition of abstract domains
- ▶ Eases development of highly expressive relational static analyses



Modular Open Platform for Static Analysis [Jou+19]

`mopsa.lip6.fr` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- ▶ Allows users to choose their composition of abstract domains
- ▶ Eases development of highly expressive relational static analyses
- ▶ Can be used as an experimentation platform



Modular Open Platform for Static Analysis [Jou+19]

`mopsa.lip6.fr` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- ▶ Allows users to choose their composition of abstract domains
- ▶ Eases development of highly expressive relational static analyses
- ▶ Can be used as an experimentation platform
 - Supports analysis of C, Python
OCaml WIP, cf. Milla Valnet's talk at TPSA (14:22)



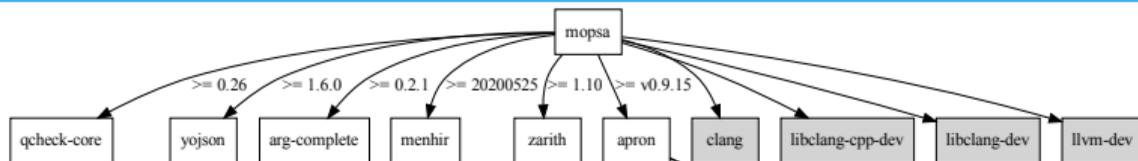
Modular Open Platform for Static Analysis [Jou+19]

`mopsa.lip6.fr` or `opam install mopsa`

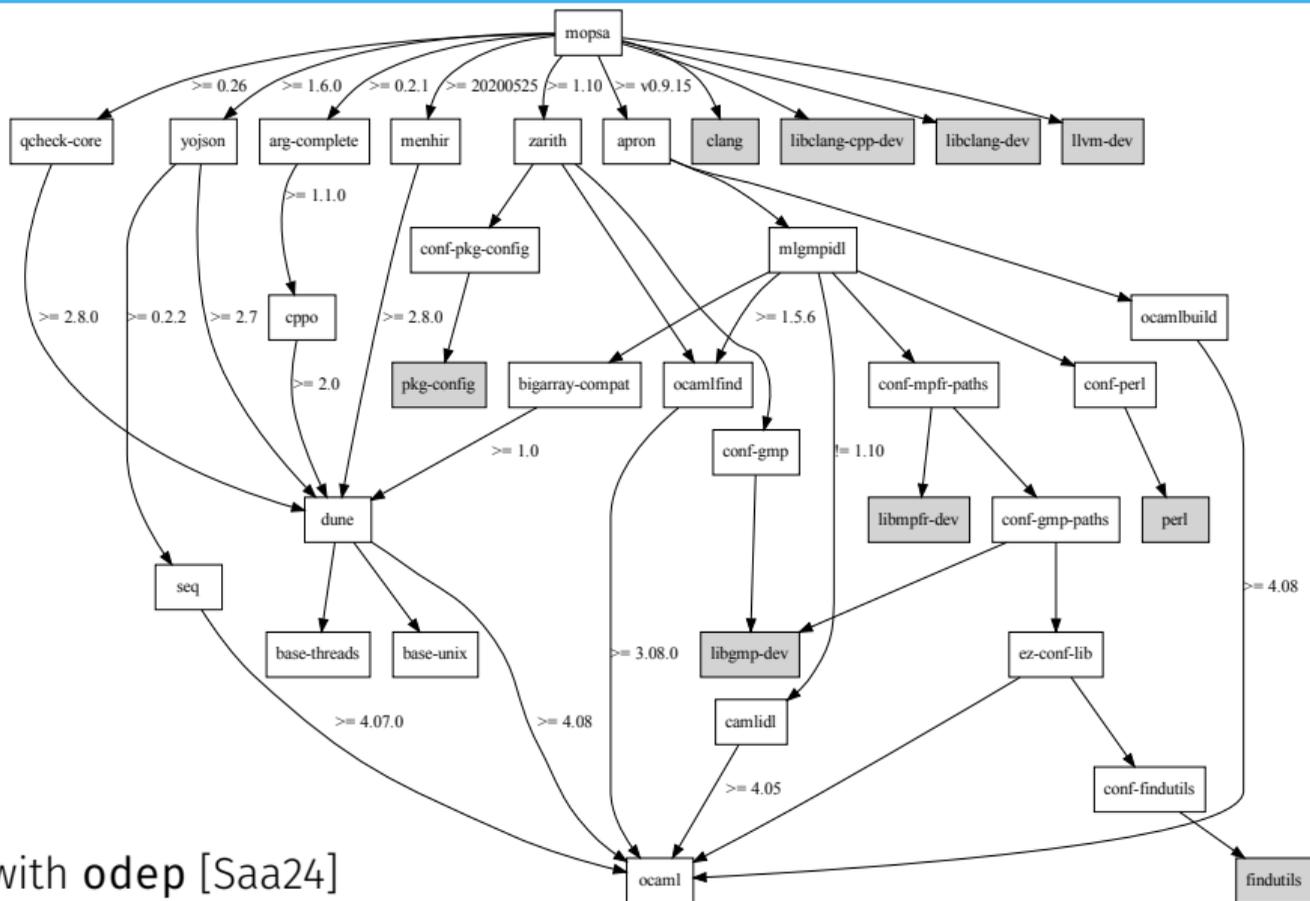
Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- ▶ Allows users to choose their composition of abstract domains
- ▶ Eases development of highly expressive relational static analyses
- ▶ Can be used as an experimentation platform
 - Supports analysis of C, Python
OCaml WIP, cf. Milla Valnet's talk at TPSA (14:22)
 - Academically competitive on real-world benchmarks (SV-Comp)

Static analyzers are complex pieces of software!



Static analyzers are complex pieces of software!



Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- ▶ Zero-install

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- ▶ Zero-install
- ▶ Supports core features of Mopsa

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- ▶ Zero-install
- ▶ Supports core features of Mopsa
- ▶ Features responsive interface supporting smartphones to computers

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- ▶ Zero-install
- ▶ Supports core features of Mopsa
- ▶ Features responsive interface supporting smartphones to computers
- ▶ Scales in number of users (purely client-side)

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

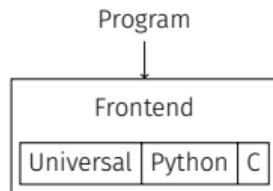
- ▶ Zero-install
- ▶ Supports core features of Mopsa
- ▶ Features responsive interface supporting smartphones to computers
- ▶ Scales in number of users (purely client-side)
- ▶ Can be convenient for onboarding or teaching

Outline

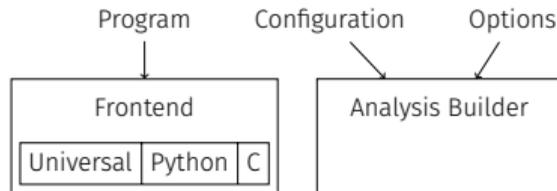
- 1 Under the Hood of Try-Mopsa
- 2 Interface Overview
- 3 Implementation Discussion
- 4 Conclusion

Under the Hood of Try-Mopsa

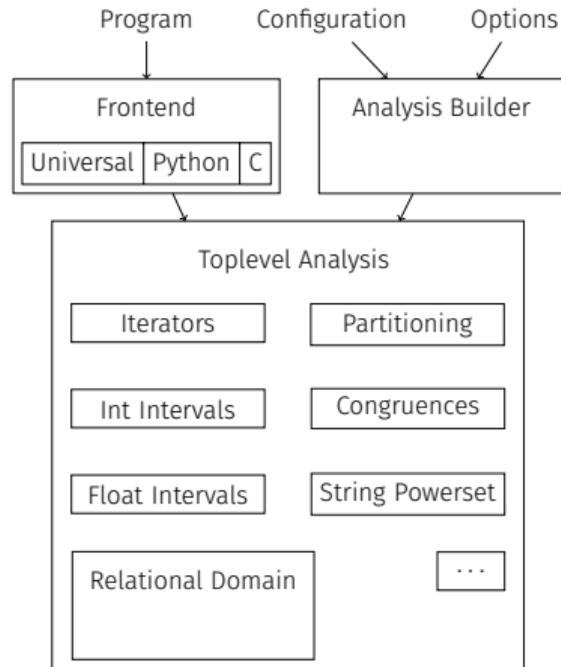
Overview of Mopsa



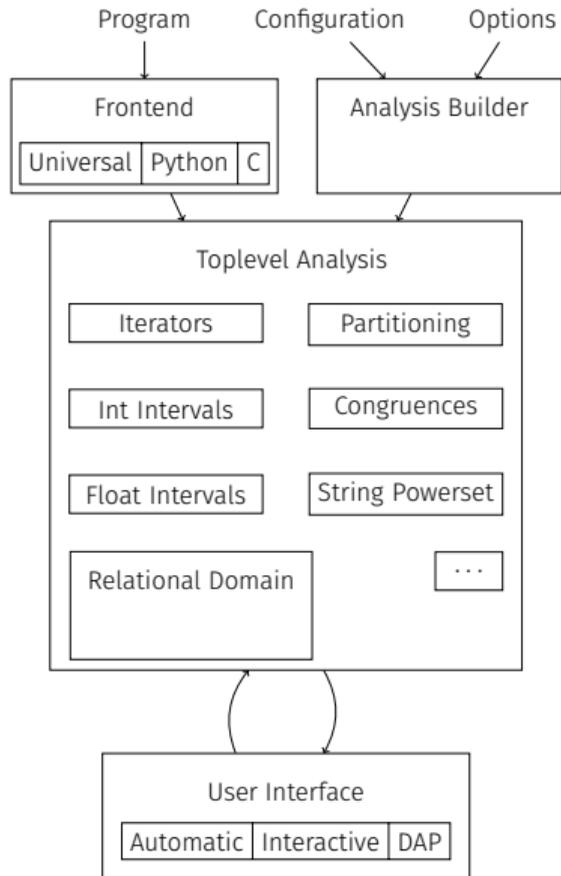
Overview of Mopsa



Overview of Mopsa



Overview of Mopsa

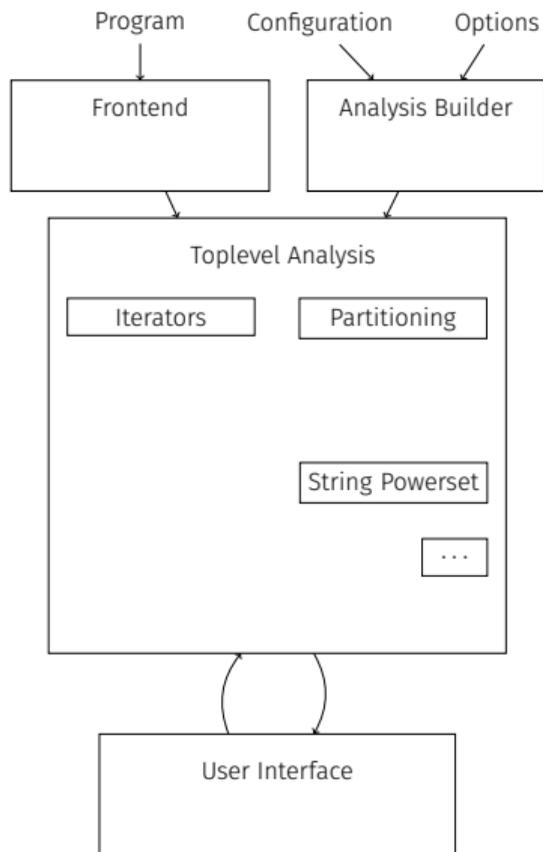


Compiling Mopsa to JavaScript

Rely on `Js_of_ocaml` [VB14]

89% of Mopsa's codebase is written in OCaml

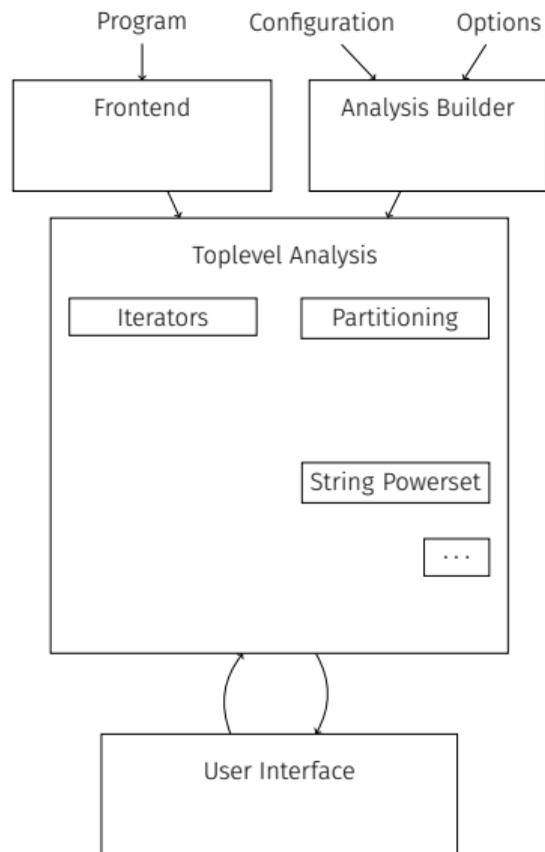
Compiling Mopsa to JavaScript



Rely on `Js_of_ocaml` [VB14]

89% of Mopsa's codebase is written in OCaml

Compiling Mopsa to JavaScript



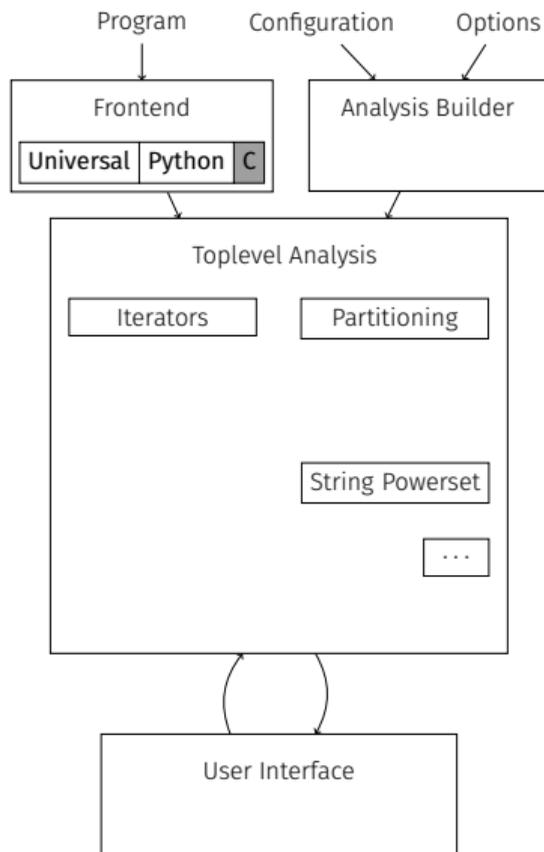
Rely on `Js_of_ocaml` [VB14]

Parsing libraries

Menhir

~~libclang~~: too much manual work for now

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

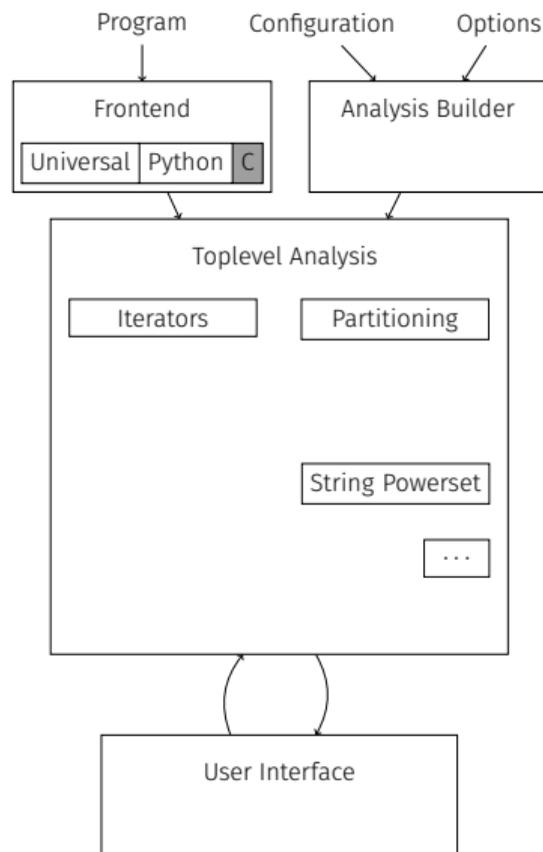
Rely on `Js_of_ocaml` [VB14]

Parsing libraries

Menhir

~~libclang~~: too much manual work for now

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

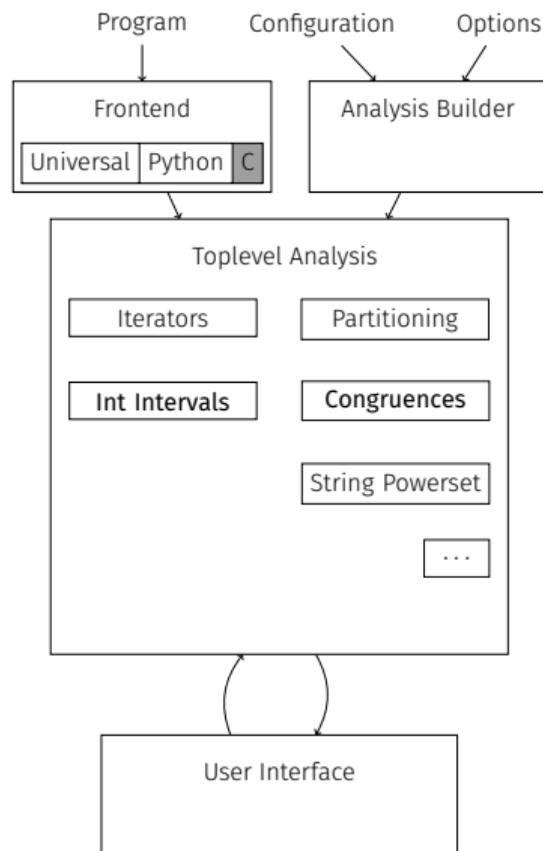
Rely on `Js_of_ocaml` [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

`Zarith` \rightsquigarrow `Zarith_stubs_js`

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

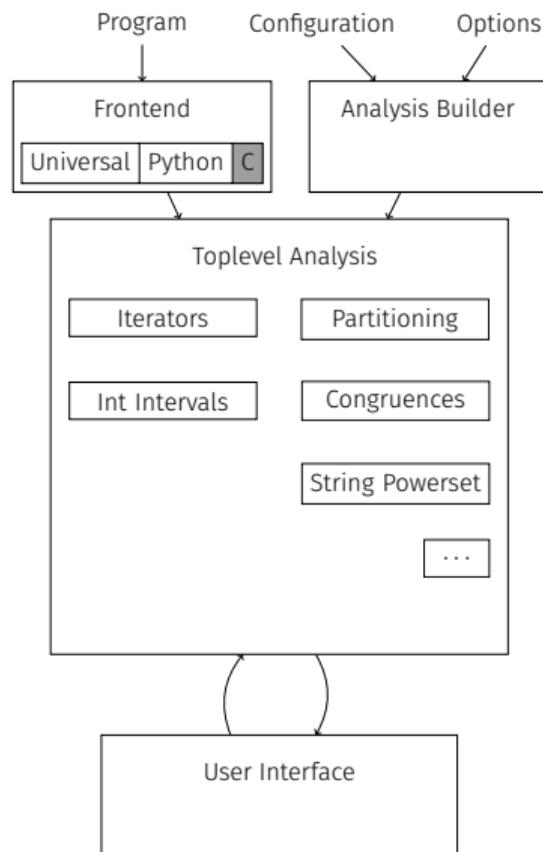
Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

`Zarith` \rightsquigarrow `Zarith_stubs_js`

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

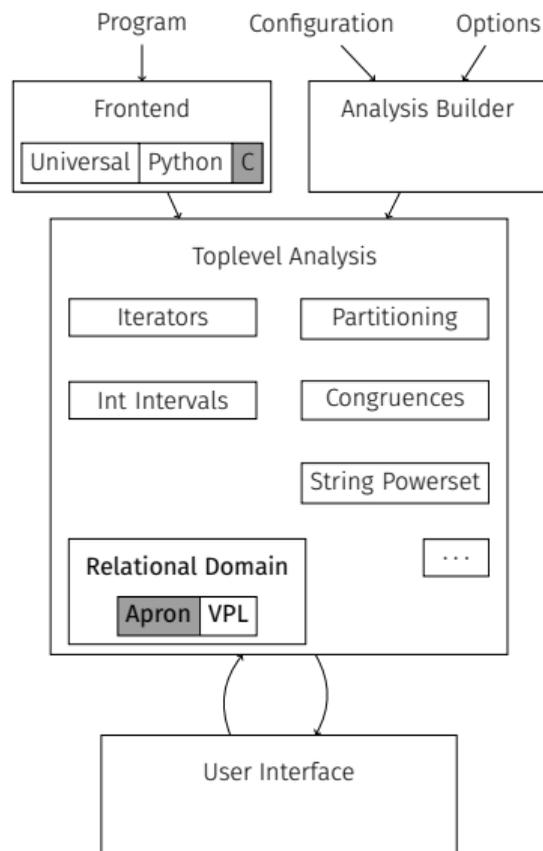
Parsing libraries

Arbitrary-precision integer arithmetic

Relational domains

Apron [JM09] \rightsquigarrow VPL [Bou+18] w/ extensions

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on `Js_of_ocaml` [VB14]

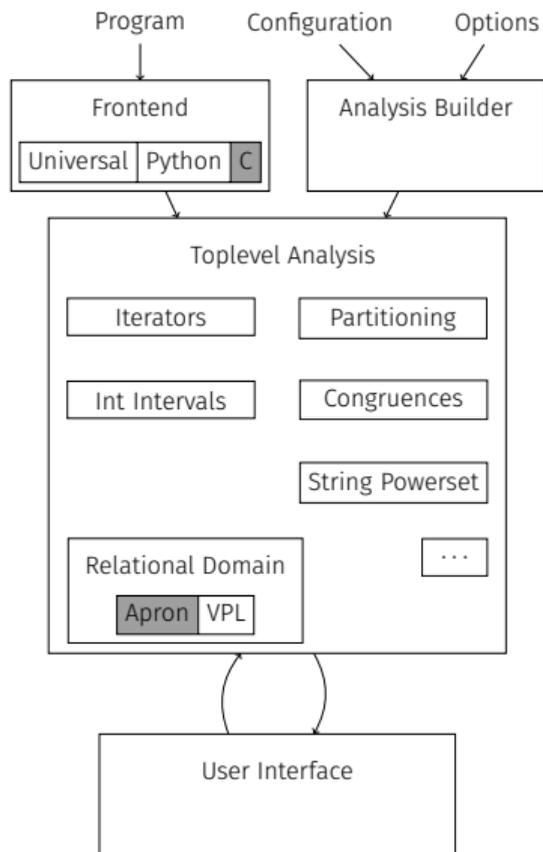
Parsing libraries

Arbitrary-precision integer arithmetic

Relational domains

`Apron` [JM09] \rightsquigarrow `VPL` [Bou+18] w/ extensions

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

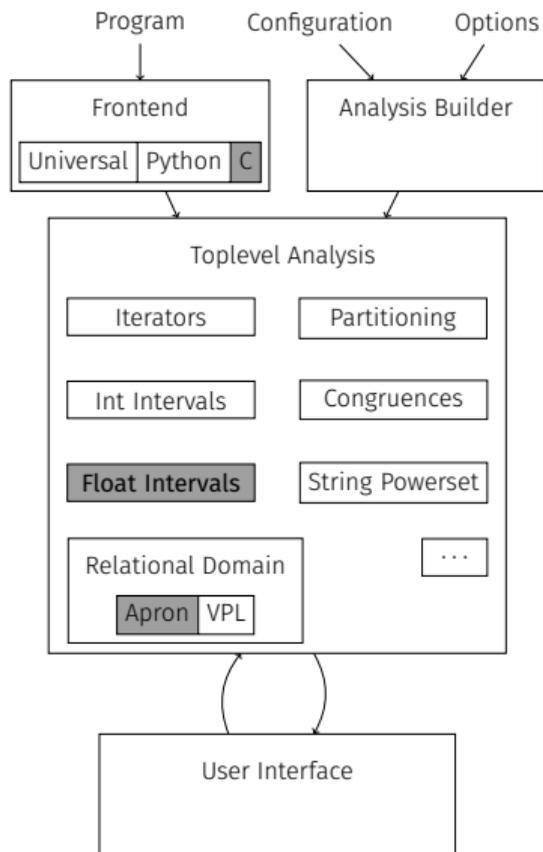
Relational domains

Floating-point

Our implementation needs to set rounding modes

Not supported in JavaScript/WebAssembly.

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

Relational domains

Floating-point

Our implementation needs to set rounding modes

Not supported in JavaScript/WebAssembly.

Toplevel

- ▶ Handles user actions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS
- ▶ Intercepts `std{in,out}`, forwards them to the toplevel

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS
- ▶ Intercepts `std{in,out}`, forwards them to the toplevel
- ▶ Relies on virtual filesystem to embed stubs and configurations

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS
- ▶ Intercepts `std{in,out}`, forwards them to the toplevel
- ▶ Relies on virtual filesystem to embed stubs and configurations

Handling Interactive User Inputs

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS
- ▶ Intercepts `std{in,out}`, forwards them to the toplevel
- ▶ Relies on virtual filesystem to embed stubs and configurations

Handling Interactive User Inputs

- ▶ Mopsa CLI: synchronous, blocking interaction loop

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS
- ▶ Intercepts `std{in,out}`, forwards them to the toplevel
- ▶ Relies on virtual filesystem to embed stubs and configurations

Handling Interactive User Inputs

- ▶ Mopsa CLI: synchronous, blocking interaction loop
- ▶ But web interfaces require asynchronous interfaces!

Handling Web Interactions

Toplevel

- ▶ Handles user actions
- ▶ Ace editor with custom highlighting and ANSI rendering
- ▶ Dynamic option generation
- ▶ Triggers/interrupts web worker

Web Worker

- ▶ Contains Mopsa in JS
- ▶ Intercepts `std{in,out}`, forwards them to the toplevel
- ▶ Relies on virtual filesystem to embed stubs and configurations

Handling Interactive User Inputs

- ▶ Mopsa CLI: synchronous, blocking interaction loop
 - ▶ But web interfaces require asynchronous interfaces!
- ⇒ Rely on `sync-message` [Moj23] (low-level synchronous comm.)

Interface Overview

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8
9 assert(0 <= j < |s|  $\implies$  s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8
9 assert(0 <= j < |s|  $\implies$  s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8
9 assert(0 <= j < |s|  $\implies$  s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8
9 assert(0 <= j < |s|  $\implies$  s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8
9 assert(0 <= j < |s|  $\implies$  s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8 // s ∈ {"a", "ab", "abc", ..., "abc...xy", "abc...xyz"}
9 assert(0 <= j < |s| ⇒ s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8 // s ∈ {"a", "ab", "abc", ..., "abc...xy", "abc...xyz"}
9 assert(0 <= j < |s| ⇒ s[j] - 'a' < |s|);
```

Running example

```
1 str s = "a";
2 int i = 1;
3 while('a' + i <= 'z') {
4     if (rand(0, 1)) break;
5     s = s @ to_string('a' + i);
6     i = i + 1;
7 }
8 // s ∈ {"a", "ab", "abc", ..., "abc...xy", "abc...xyz"}
9 assert(0 <= j < |s| ⇒ s[j] - 'a' < |s|);
```

⇒ `try-mopsa.rmonat.fr`

Implementation Discussion

- ▶ Mopsa \simeq 87kLOC

Implementation

- ▶ Mopsa \simeq 87kLOC
- ▶ Try-Mopsa

Implementation

- ▶ Mopsa \simeq 87kLOC
- ▶ Try-Mopsa
 - 1,500 lines of HTML/CSS/JS

- ▶ Mopsa \simeq 87kLOC
- ▶ Try-Mopsa
 - 1,500 lines of HTML/CSS/JS
 - 670 lines of OCaml (toplevel/worker)

- ▶ Mopsa \simeq 87kLOC
- ▶ Try-Mopsa
 - 1,500 lines of HTML/CSS/JS
 - 670 lines of OCaml (toplevel/worker)
 - 500 lines of OCaml (VPL)

- ▶ Mopsa \simeq 87kLOC
- ▶ Try-Mopsa
 - 1,500 lines of HTML/CSS/JS
 - 670 lines of OCaml (toplevel/worker)
 - 500 lines of OCaml (VPL)
 - Compiled, optimized size: 3.1 megabytes

- ▶ Mopsa \simeq 87kLOC
- ▶ Try-Mopsa
 - 1,500 lines of HTML/CSS/JS
 - 670 lines of OCaml (toplevel/worker)
 - 500 lines of OCaml (VPL)
 - Compiled, optimized size: 3.1 megabytes
 - *No breaking changes*

Performance

| Program | Binary execution | Firefox execution |
|------------------|------------------|-------------------|
| attributes.py | 0.09s \pm 0.00 | 0.52s \pm 0.03 |
| fspath.py | 0.09s \pm 0.00 | 0.49s \pm 0.02 |
| list.py | 0.15s \pm 0.01 | 0.88s \pm 0.03 |
| loop.py | 0.12s \pm 0.01 | 0.72s \pm 0.03 |
| recency.py | 0.08s \pm 0.01 | 0.63s \pm 0.03 |
| str_alphabet.u | 0.04s \pm 0.01 | 0.20s \pm 0.01 |
| str_alphabet2.u | 0.23s \pm 0.01 | 0.81s \pm 0.01 |
| str_conc_loop.u | 0.08s \pm 0.01 | 0.29s \pm 0.01 |
| str_conc_loop2.u | 0.04s \pm 0.01 | 0.15s \pm 0.01 |

Performance

| Program | Binary execution | Firefox execution |
|------------------|------------------|-------------------|
| attributes.py | 0.03s ± 0.01 | 0.15s ± 0.01 |
| fspath.py | 0.03s ± 0.01 | 0.15s ± 0.01 |
| list.py | 0.03s ± 0.01 | 0.15s ± 0.01 |
| loop.py | 0.03s ± 0.01 | 0.15s ± 0.01 |
| recenc.py | 0.03s ± 0.01 | 0.15s ± 0.01 |
| str_alphabet.u | 0.04s ± 0.01 | 0.20s ± 0.01 |
| str_alphabet2.u | 0.23s ± 0.01 | 0.81s ± 0.01 |
| str_conc_loop.u | 0.08s ± 0.01 | 0.29s ± 0.01 |
| str_conc_loop2.u | 0.04s ± 0.01 | 0.15s ± 0.01 |

- ▶ 5× slowdown compared to native binary
- ▶ Raw performance is not a priority
- ▶ Still reasonable analysis times

Playwright framework to assess the browser compatibility of Try-Mopsa

Conformance tests

5 tests x (3 desktop browsers + 2 mobile browsers x 2 viewports)

Identified several rendering issues on mobile

Playwright framework to assess the browser compatibility of Try-Mopsa

Conformance tests

5 tests x (3 desktop browsers + 2 mobile browsers x 2 viewports)

Identified several rendering issues on mobile

Additional tests

Universal: 13 programs x 11 configurations

Python: 5 programs x 6 configurations

Browser compatibility (II)

Playwright Test

PLAYWRIGHT

Filter (e.g. text, @tag)

Status: all Projects: chromium firefox webkit Mobile Chrome Mobile C...

7/7 passed (100%)

- str_agmaer.u string_product_relaional.json
- str_alphabet2.u string_product_relational.json
- str_conc_loop.u intervals.json
- str_conc_loop.u string_product_relational.json
- str_conc_loop2.u string_product_relational.json
- str_dot_at_end.u string_product_relational.json
- str_double.u intervals.json
- str_double.u string_product_relational.json
- str_double2.u intervals.json
- str_double2.u string_product_relational.json
- attributes.py values-relational.json
- attributes.py values.json
- fspath.py values-relational.json
- fspath.py values.json
- list.py values-relational.json
- list.py values.json
- loop.py values-relational.json
- loop.py values.json
- recency.py values-relational.json
- recency.py values.json
- conformance.spec.ts
 - universal_default_tv
 - universal_default_rel
 - universal_default_rel_interactive 49.8s
 - chromium 6.0s
 - Firefox 8.0s
 - webkit 8.0s
 - Mobile Chrome 6.1s
 - Mobile Chrome Landscape 5.7s
 - Mobile Safari 8.2s
 - Mobile Safari Landscape 8.2s
 - error_message
 - share_button

Actions Metadata

✓ Passed 9.3s

> Before Hooks 3.1s

Navigate to ""

Select option locator("#engine_select")

Select option locator("#config_selector") 121ms

Click locator("#btn-run") 182ms

Wait for function 822ms

Press "b" 48ms

Press "" 40ms

Press "4" 38ms

Press "." 38ms

Press "" 33ms

Press "c" 40ms

Press ";" 33ms

Press "" 41ms

Press "c" 39ms

Press "." 37ms

Press "" 36ms

Press "c" 49ms

Press "." 47ms

Press "" 50ms

Press "c" 70ms

Locator Source Call Log Errors Console 162 Network 49 Attachments Annotations

http://localhost:3000/

Editor Configuration Options Guide

Share Run code

Code Editor Universal Python Select file

```
1 int i = 0;
2 int j = rand(10, 20);
3 while (i < j) {
4     i = i + 1;
5 }
6 print();
7 assert(i == j);
8
```

Mopsa Analysis

```
25 7 assert(i == j);
26 8
27 1 i = (i + 1);
28 Input.u:4.4-14
29 1 int i = 0;
30 2 int j = rand(10, 20);
31 3 while (i < j) {
32 4     i = i + 1;
33 5 }
34 6 print();
35 7 assert(i == j);
36 8
37 1 i = (i + 1);
38 Input.u:4.4-14
39 1 int i = 0;
40 2 int j = rand(10, 20);
41 3 while (i < j) {
42 4     i = i + 1;
43 5 }
44 6 print();
45 7 assert(i == j);
46 8
47 1 i = (i + 1);
48 int-1tv U strings :
49 i : [1,19] ,
50 numeric-relations : [ i <= i, 10 < j, j <= 20, i + 1 <= j ]
51 mopsa
```

No errors

Conclusion

- ▶ Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
Less adaptation required, but maintenance and security issues

- ▶ Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
Less adaptation required, but maintenance and security issues
- ▶ Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25], Binsec tutorial [RB25]
Try-Mopsa supports relational domains and interactive user input

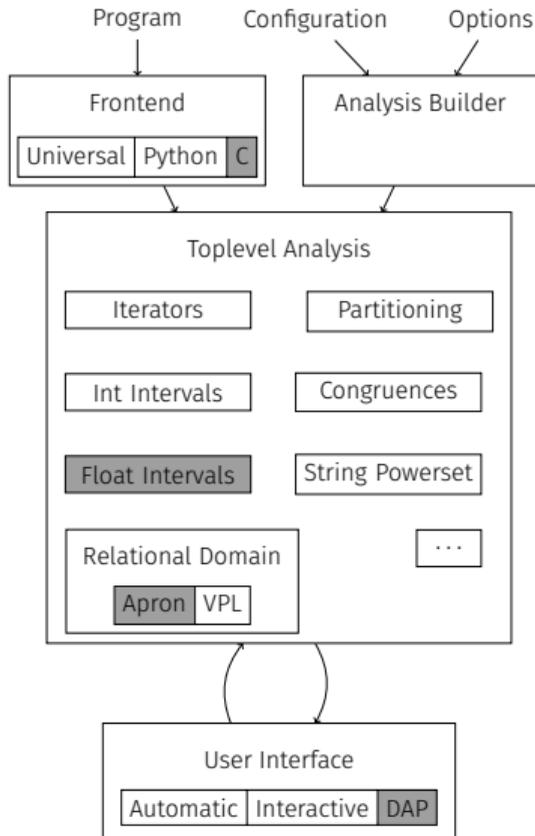
- ▶ Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
Less adaptation required, but maintenance and security issues
- ▶ Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25], Binsec tutorial [RB25]
Try-Mopsa supports relational domains and interactive user input
- ▶ LiSA for teaching [Neg+24]

- ▶ Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
Less adaptation required, but maintenance and security issues
- ▶ Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25], Binsec tutorial [RB25]
Try-Mopsa supports relational domains and interactive user input
- ▶ LiSA for teaching [Neg+24]
- ▶ Interaction with IDEs [LDB19]

- ▶ Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
Less adaptation required, but maintenance and security issues
- ▶ Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25], Binsec tutorial [RB25]
Try-Mopsa supports relational domains and interactive user input
- ▶ LiSA for teaching [Neg+24]
- ▶ Interaction with IDEs [LDB19]
- ▶ Works between concrete and abstract debuggers [Do+20; Hol+24; MVR23]

- ▶ Pure JS version of Mopsa

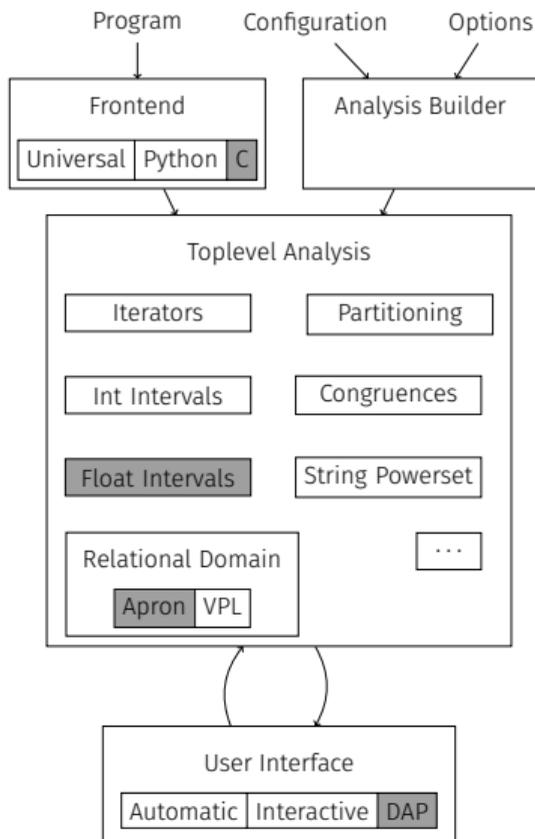
Conclusion



Gray = unsupported by Try-Mopsa.

- ▶ Pure JS version of Mopsa
- ▶ Supports polyhedra, abstract debugger

Conclusion



Gray = unsupported by Try-Mopsa.

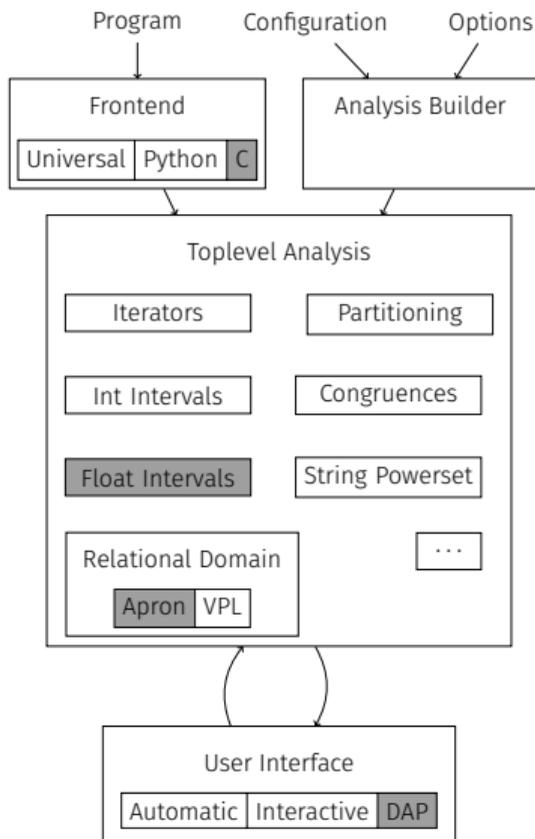
- ▶ Pure JS version of Mopsa
- ▶ Supports polyhedra, abstract debugger
- ▶ Works on smartphones and desktops alike

```
1- str to_string(int i) {
2   str r = "a";
3   r[0] = i;
4   return r;
5 }
6 str s = "a";
7 int i = 1;
8 int j;
9 while('a' + i <= 'z') {
10  if (rand(0, 1)) break;
11  s = s @ to_string('a' + i);
12  i = i + 1;
13 }
14 print();
15 j = rand(0, |s|-1);
16 // For all 0 <= j < |s|, s[j] - 'a' < |s|
17 // which means s in {a, (a+b)^2, (a+b+c)^3, ...}
18 assert(s[j] - 'a' < |s|);
19
```

Mopsa Analysis

```
1 input.u:14.0-8
2 cur :
3 int-ityv : |
4   i - [1,26],
5   j - [-,+=],
6   len(s) - [1,26],
7   ord(s) - [97,122],
8   numeric-relations :
9   [ len(s) = i, ord(s) ≤ 96 + i, 1 ≤
10    t, i ≤ 26 ]
11 Analysis terminated successfully
12 (with assumptions)
13 ✓ No alarm
14 Analysis time: 0.376s
15 Checks summary: 3 total, ✓
16   3 safe (selectivity: 100.00%)
17   Subscript access: 2 total, ✓ 2
18     safe
19   Assertion failure: 1 total, ✓ 1
20     safe
```

Conclusion



Gray = unsupported by Try-Mopsa.

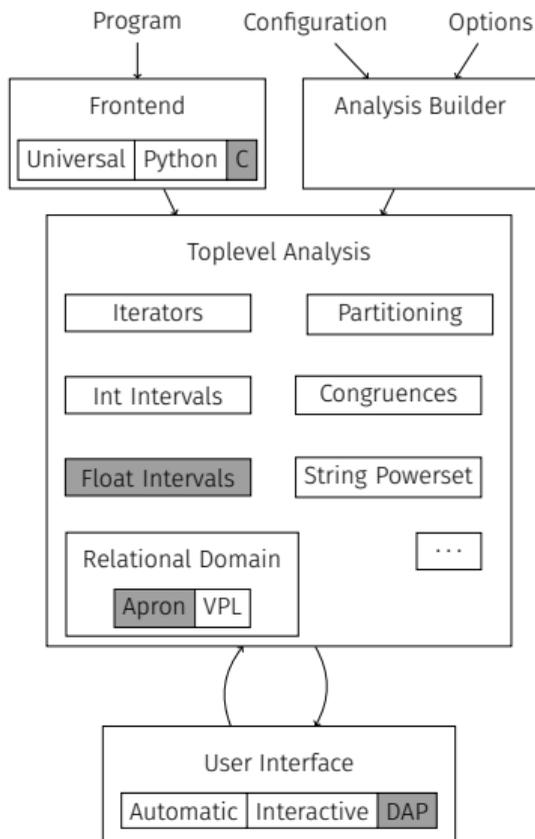
- ▶ Pure JS version of Mopsa
- ▶ Supports polyhedra, abstract debugger
- ▶ Works on smartphones and desktops alike
- ▶ Useful for demo or lightweight teaching purposes

```
1- str to_string(int i) {
2-   str r = "a";
3-   r[0] = i;
4-   return r;
5- }
6- str s = "a";
7- int i = 1;
8- int j;
9- while('a' + i <= 'z') {
10-  if (rand(0, 1)) break;
11-  s = s @ to_string('a' + i);
12-  i = i + 1;
13- }
14- print();
15- j = rand(0, |s|-1);
16- // For all 0 <= j < |s|, s[j] - 'a' < |s|
17- // which means s in {a, (a+b)^2, (a+b+c)^3, ...}
18- assert(s[j] - 'a' < |s|);
19- }
```

Mopsa Analysis

```
1 input.u:14.0-8
2 cur :
3 int-ity : |
4 i - [1,26],
5 j - [-,+=],
6 len(s) - [1,26],
7 ord(s) - [97,122] ,
8 numeric-relations :
9 [ len(s) = i, ord(s) ≤ 96 + i, 1 ≤ i, i ≤ 26 ]
10 Analysis terminated successfully
    (with assumptions)
11 ✓ No alarm
12 Analysis time: 0.376s
13 Checks summary: 3 total, ✓
    3 safe (selectivity: 100.00%)
14 Subscript access: 2 total, ✓ 2
    safe
15 Assertion failure: 1 total, ✓ 1
    safe
```

Conclusion



Gray = unsupported by Try-Mopsa.

- ▶ Pure JS version of Mopsa
- ▶ Supports polyhedra, abstract debugger
- ▶ Works on smartphones and desktops alike
- ▶ Useful for demo or lightweight teaching purposes

`try-mopsa.rmonat.fr`

The screenshot shows the Mopsa analyzer interface with a code editor and an analysis output window.

```
1- str to_string(int i) {
2   str r = "a";
3   r[0] = i;
4   return r;
5 }
6 str s = "a";
7 int i = 1;
8 int j;
9 while('a' + i <= 'z') {
10  if (rand(0, 1)) break;
11  s = s @ to_string('a' + i);
12  i = i + 1;
13 }
14 print();
15 j = rand(0, |s|-1);
16 // For all 0 <= j < |s|, s[j] - 'a' < |s|
17 // which means s in {a, (a+b)^2, (a+b+c)^3, ...}
18 assert(s[j] - 'a' < |s|);
19
```

Mopsa Analysis

```
1 input.u:14.0-8
2 cur :
3 int-ity : |
4   i - [1,26],
5   j - [-,+=],
6   len(s) - [1,26],
7   ord(s) - [97,122],
8   numeric-relations :
9   [ len(s) = i, ord(s) ≤ 96 + i, 1 ≤
    i, i ≤ 26 ]
10 Analysis terminated successfully
    (with assumptions)
11 ✓ No alarm
12 Analysis time: 0.376s
13 Checks summary: 3 total, ✓
    3 safe (selectivity: 100.00%)
14 Subscript access: 2 total, ✓ 2
    safe
15 Assertion failure: 1 total, ✓ 1
    safe
```

References – I

- [Bou+18] Sylvain Boulmé et al. **“The Verified Polyhedron Library: an Overview”**. In: IEEE, 2018, pp. 9–17. DOI: [10.1109/SYNASC.2018.00014](https://doi.org/10.1109/SYNASC.2018.00014).
- [Do+20] Lisa Nguyen Quang Do et al. **“Debugging Static Analysis”**. In: IEEE Trans. Software Eng. 7 (2020), pp. 697–709. DOI: [10.1109/TSE.2018.2868349](https://doi.org/10.1109/TSE.2018.2868349).
- [Hol+24] Karoliine Holter et al. **“Abstract Debuggers: Exploring Program Behaviors using Static Analysis Results”**. In: ed. by Jonathan Edwards and Marcel Taeumel. ACM, 2024, pp. 130–146. DOI: [10.1145/3689492.3690053](https://doi.org/10.1145/3689492.3690053).

References – II

- [Jea09] Bertrand Jeannet. Web Interface for the Interproc Analyzer. 2009. URL: <https://pop-art.inrialpes.fr/interproc/interprocweb.cgi.html>.
- [JM09] Bertrand Jeannet and Antoine Miné. **“Apron: A Library of Numerical Abstract Domains for Static Analysis”**. In: Springer, 2009, pp. 661–667.
- [Jou+19] M. Journault et al. **“Combinations of reusable abstract domains for a multilingual static analyzer”**. In: New York, USA, July 2019, pp. 1–17.

References – III

- [LDB19] Linghui Luo, Julian Dolby, and Eric Bodden. **“MagpieBridge: A General Approach to Integrating Static Analyses into IDEs and Editors (Tool Insights Paper)”**. In: ed. by Alastair F. Donaldson. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 21:1–21:25. DOI: [10.4230/LIPICS.ECOOP.2019.21](https://doi.org/10.4230/LIPICS.ECOOP.2019.21).
- [LM25] Pierre Lermusiaux and Benoît Montagu. **Web Demo for the Salto Analyzer**. 2025. URL: <https://salto.gitlabpages.inria.fr/demo/salto.html>.
- [Min12] Antoine Miné. **Web Interface for Sufficient Condition Polyhedral Prototype Analyzer**. 2012. URL: <https://mine.perso.lip6.fr/banal/>.

References – IV

- [Moj23] Alex Mojaki. [sync-message](#). Version 0.0.12. Oct. 2023.
- [MPH22] Jose F. Morales, Guillermo García Pradales, and Manuel Hermenegildo. [Ciao Prolog Playground](#). 2022. URL: <https://ciao-lang.org/playground/>.
- [MVR23] Mats Van Molle, Bram Vandenbogaerde, and Coen De Roover. **“Cross-Level Debugging for Static Analysers”**. In: ed. by João Saraiva, Thomas Degueule, and Elizabeth Scott. ACM, 2023, pp. 138–148. DOI: [10.1145/3623476.3623512](https://doi.org/10.1145/3623476.3623512).

References – V

- [Neg+24] Luca Negrini et al. **“Teaching Through Practice: Advanced Static Analysis with LiSA”**. In: ed. by Emil Sekerinski and Leila Ribeiro. Lecture Notes in Computer Science. Springer, 2024, pp. 43–57. DOI: [10.1007/978-3-031-71379-8_3](https://doi.org/10.1007/978-3-031-71379-8_3).
- [RB25] Frédéric Recoules and Sébastien Bardin. **BINSEC Tutorial at PLDI’25**. 2025. URL: <https://binsec.github.io/tutorial-pldi2025/>.
- [Saa24] Simmo Saan. **Odep: Dependency graphs for OCaml modules, libraries and packages**. Version 0.2.1. Apr. 2024.
- [Urb15] Caterina Urban. **Web Interface for FuncTion**. 2015. URL: <https://www.di.ens.fr/~urban/FuncTion.html>.

References – VI

- [VB14] Jérôme Vouillon and Vincent Balat. **“From bytecode to JavaScript: the Js_of_ocaml compiler”**. In: Softw. Pract. Exp. 8 (2014), pp. 951–972. DOI: 10.1002/SPE.2187.