

# Verificarea Monocypher cu Framac/Eva

Adrian Manea

SLA, 410

FramaC — implementat în OCaml, pentru verificarea programelor C.  
Oferă o verificare automată sau cu adnotări în ACSL.  
Face doar verificări „de bază”. Extins prin plugin-uri:

- WP = Weakest Precondition calculus;
- RTE = RunTime Evaluation;
- Eva = Evolved Value Analysis;
- poate integra demonstratorul Coq.

Detalii: [FramaC, 2019a], [Baudin, 2019], [FramaC, 2019b].

# Exemplu: FramaC CLI + GUI

The screenshot displays the FramaC GUI interface. On the left, the CLI window shows the following commands and output:

```
acsl-ex git:(master) x frama-c abs.c -save abs.sav
[kernel] Parsing abs.c (with preprocessing)
acsl-ex git:(master) x frama-c-gui -load abs.sav &!
acsl-ex git:(master) x cat abs.c
/*
 * ensures positive_value: function_result: \result >= 0;
 * ensures (val >= 0 ==> \result == val) &&
 * (val < 0 ==> \result == -val);
 */
int abs(int val) {
  if (val < 0)
    return -val;
  return val;
}
acsl-ex git:(master) x
```

On the right, the code editor shows the 'abs.c' file with the following code:

```
1 /*
2  * ensures positive_value: function_result: \result >= 0;
3  * ensures (val >= 0 ==> \result == val) &&
4  * (val < 0 ==> \result == -val);
5  */
6 int abs(int val) {
7   if (val < 0)
8     return -val;
9   return val;
10 }
11
```

The bottom panel shows the 'WP' (Weakest Precondition) tab with the following settings:

- Model... Typed
- Script... (None)
- Provers... Alt-Ergo (native)
- RTE ☐ Split ☐ Trace ☐
- Steps 0
- Depth 0
- Timeout 10
- Process 4
- Occurrence: Current var: None, Enable ☐ Follow focus ☐ Read ☒ Write ☐
- Metrics: Launch
- Impact: Erase

Bazat pe **interpretarea abstractă** ([Cousot, 2008]).

Ideea: pornind de la *semantica concretă* a unei bucăți de cod, se *abstractizează*, pentru ușurarea calculelor.

Eva calculează valorile și intervalele valorilor unor variabile.

Face sub- sau supra-aproximări, deci produce *alarme*, nu erori.

Este o bibliotecă criptografică simplă și rapidă. ([LoupVaillant, 2019a])

Folosește algoritmi criptografici clasici: sha\_512, ChaCha20 ([LoupVaillant, 2019b], [LoupVaillant, 2017]).

# Rularea inițială

```
# rezultate stdout:  
$ frama-c *.c  
# rezultate log  
$ frama-c *.c > log ; cat log | less  
# rezultate salvate, apoi GUI  
$ frama-c *.c -save firstrun.sav  
$ frama-c-gui -load firstrun.sav
```

Se pot da opțiuni suplimentare de preprocesor și se încarcă în GUI rezultatul, pentru a vizualiza mai bine rezultatele.

# Rularea inițială în GUI

The screenshot displays the initial state of a GUI for running a program. It is divided into several panels:

- Left Panel (Project Explorer):** Lists files and folders including `diff`, `ed25519_key`, `ed25519_sign1`, `ed25519_sign2`, `equal`, `generic_test`, `hchacha20`, `is_digit`, `iterate_x25519`, `key_exchange`, `main` (highlighted), `next_number`, `poly1305`, `read_hex_line`, and `sha512`.
- Top Middle Panel (Source Code):** Shows the `main` function in `test.c`. It includes variable declarations for `tmp` (0-15), `status`, and `char` arrays for keys and signatures. It calls `generic_test` for various operations like `equal`, `diff`, `chacha20`, and `blake2b`.
- Top Right Panel (Test Code):** Shows the `test` function in `test.c`. It performs cryptographic operations like `crypto_lock`, `crypto_unlock`, `crypto_memcmp`, and `crypto_read` on a `box` array, comparing results with expected values.
- Bottom Panel (Console):** Displays a message from `test.c:451` indicating a `varadic` call to `printf` with a non-static format argument. The message is: `no specification will be generated.`

```
$ frama-c -load parsed.sav -val -val-builtins-auto \  
-no-val-show-progress -memexec-all -save value.sav > log
```

- -val = Eva;
- -val-builtins-auto = verificările standard;
- -no-val-show-progress = nu se afișează mesaje cînd se intră în fiecare funcție;
- -memexec-all = se păstrează în cache valorile calculate, pentru refolosire.



```
[value] Analyzing a complete application starting at main
[value] Computing initial state
[value] Initial state computed
[value:initial-state] Values of globals at initialization
__fc_errno in [--..--]
__fc_stderr in {{ NULL ; &S__fc_stderr[0] }}
__fc_strtok_ptr in {0}
K[0] in {4794697086780616226}
[1] in {8158064640168781261}
...
blake2b_compress_sigma[0][0] in {0}
[0][1] in {1}
...
[value:alarm] test.c:126: Warning:
out of bounds write. assert \valid(v->buf + v->size);
```

```
[value:alarm] test.c:121: Warning:  
function memcpy: precondition 'valid_src' got status unknown.  
[value:alarm] monocypher.c:839: Warning:  
signed overflow. assert g3 * 19 leq 2147483647;  
[kernel] monocypher.c:494:  
more than 200(253) locations to update in array. Approximating.
```

# Rafinarea verificării în GUI

The screenshot displays the Monocypher GUI interface, which is divided into several sections:

- Left Panel:** Contains a file explorer with a list of files including `chacha20_rounds`, `copy_block`, `crypto_aead_lock`, `crypto_aead_unlock`, `crypto_argon2i`, `crypto_blake2b`, `crypto_blake2b_final`, `crypto_blake2b_general`, `crypto_blake2b_general_init`, `crypto_blake2b_init`, `crypto_blake2b_update`, `crypto_chacha20_encrypt`, `crypto_chacha20_H`, `crypto_chacha20_init`, `crypto_chacha20_stream`, and `crypto_chacha20_xinit`. Below this is a section for **WP** (Work Package) with fields for **Model...** and **Typed**, and a **Script...** dropdown set to **(None)**. There are also checkboxes for **Provers** (set to **Alt-Ergo (native)**) and **RTE** (set to **Split**).
- Top Panel:** Displays the source code for `chacha20_rounds` and `test.c`. The `chacha20_rounds` code includes assertions for initialization and various cryptographic operations. The `test.c` code defines a `monocypher_c` function that performs a series of operations on a `u32` array.
- Bottom Panel:** Contains a table of messages generated during the verification process. The table has columns for **Kind**, **Source**, **Plugin**, and **Message**. The messages are categorized by **monocypher\_c** and include various assertions and warnings.

The **Message Log** shows the following entries:

Kind	Source	Plugin	Message
monocypher_c	104	value	accessing uninitialized left-value. assert Initialized(out + 5);
monocypher_c	104	value	accessing uninitialized left-value. assert Initialized(out + 13);
monocypher_c	104	value	accessing uninitialized left-value. assert Initialized(out + 9);
monocypher_c	105	value	accessing uninitialized left-value. assert Initialized(out + 2);
monocypher_c	105	value	accessing uninitialized left-value. assert Initialized(out + 6);
monocypher_c	105	value	accessing uninitialized left-value. assert Initialized(out + 14);
monocypher_c	105	value	accessing uninitialized left-value. assert Initialized(out + 10);
monocypher_c	106	value	accessing uninitialized left-value. assert Initialized(out + 3);
monocypher_c	106	value	accessing uninitialized left-value. assert Initialized(out + 7);
monocypher_c	106	value	accessing uninitialized left-value. assert Initialized(out + 15);

# Rafinarea verificării în GUI

```
vec_del(& expected);
{
    size_t i_0 = (unsigned int)0;
    while (i_0 < nb_vectors) {
        {
            vec_del(inputs + i_0);
        }
        i_0 += (size_t)1;
    }
}
/* assert Value: signed overflow: nb_tests + 1 > 2147483647; */
nb_tests ++;
}
}
/* sequence */
;
{
    if (status != 0) {
        tmp_4 = "FAILED";
    }
    else {
        tmp_4 = "OK";
    }
}
}
printf va 2("%s %3d tests: %s\n", (char *)tmp_4, nb_tests, (char *)filename);
free(void *inputs);
stream_close(stream &);
Surely valid; verified (including all of its dependencies)
```

```
test.c monocypher.c
status |= vec_cmp(&output, &expected);
208
209
210 vec_del(&output);
211 vec_del(&expected);
212 FOR (i, 0, nb_vectors) { vec_del(inputs + i); }
213 nb_tests++;
214 }
215 printf("%s %3d tests: %s\n",
216 status != 0 ? "FAILED" : "OK", nb_tests, filename);
217 free(inputs);
218 stream_close(&stream);
219 return status;
220 }
221
222 //////////////////////////////////////
223 // test the test suite //
224 //////////////////////////////////////
225 static int equal(const vector v[]) { return vec_cmp(v, v + 1); }
226 static int diff (const vector v[]) { return !vec_cmp(v, v + 1); }
227
228 //////////////////////////////////////
229 // the tests proper //
230 //////////////////////////////////////
231 sv chacha20(const vector in[], vector *out)
232 {
233     const vector *key = in;
234     const vector *nonce = in + 1;
235     crypto_chacha_ctx ctx;
```

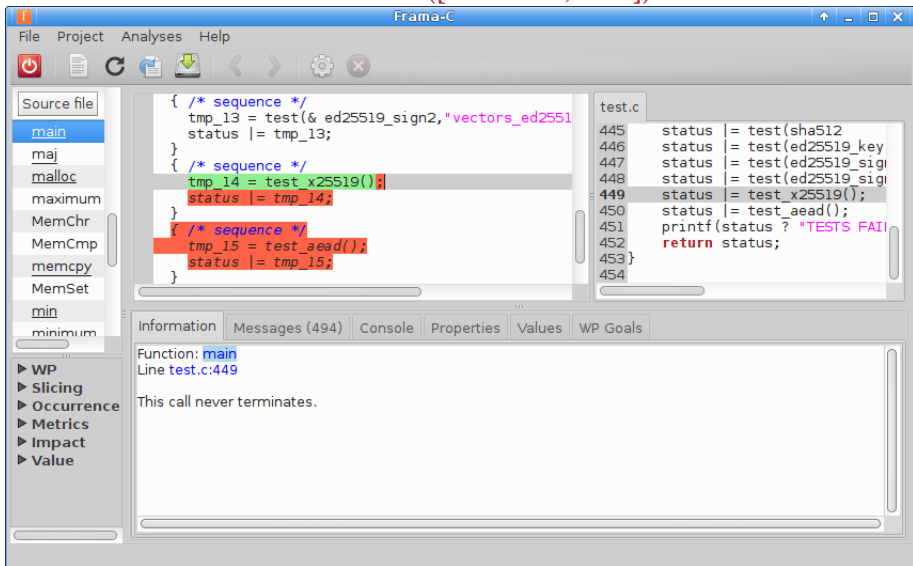
Information Messages (948) Console Properties Values Red Alarms WP Goals

☒ Multiple selections ☐ Expand rows ☒ Consolidated value ☒ Per callstack

Selection	Callstack	inputs	inputs (after)	(void *)inputs	(void *)inputs (after)
	all	{{ (vector *)\$6_malloc_alloc_122_5; (vector *)\$6_malloc_alloc_122_11; (vector *)\$6_malloc_alloc_122_17; (vector *)\$6_malloc_alloc_122_23; (vector *)\$6_malloc_alloc_122_29; (vector *)\$6_malloc_alloc_122_35; (vector *)\$6_malloc_alloc_122_41; (vector *)\$6_malloc_alloc_122_48; (vector *)\$6_malloc_alloc_122_54; (vector *)\$6_malloc_alloc_122_60; (vector *)\$6_malloc_alloc_122_66; (vector *)\$6_malloc_alloc_122_72; (vector *)\$6_malloc_alloc_122_78 }}	ESCAPINGADDR	{{ (void *)\$6_malloc_alloc_122_5; (void *)\$6_malloc_alloc_122_11; (void *)\$6_malloc_alloc_122_17; (void *)\$6_malloc_alloc_122_23; (void *)\$6_malloc_alloc_122_29; (void *)\$6_malloc_alloc_122_35; (void *)\$6_malloc_alloc_122_41; (void *)\$6_malloc_alloc_122_48; (void *)\$6_malloc_alloc_122_54; (void *)\$6_malloc_alloc_122_60; (void *)\$6_malloc_alloc_122_66; (void *)\$6_malloc_alloc_122_72; (void *)\$6_malloc_alloc_122_78 }}	BOTTOM
	main	{{ (vector *)\$6_malloc_alloc_122_5 }}	ESCAPINGADDR	{{ (void *)\$6_malloc_alloc_122_5 }}	BOTTOM
	main	{{ (vector *)\$6_malloc_alloc_122_11 }}	ESCAPINGADDR	{{ (void *)\$6_malloc_alloc_122_11 }}	BOTTOM
	main	{{ (vector *)\$6_malloc_alloc_122_17 }}	ESCAPINGADDR	{{ (void *)\$6_malloc_alloc_122_17 }}	BOTTOM
	main	{{ (vector *)\$6_malloc_alloc_122_23 }}	ESCAPINGADDR	{{ (void *)\$6_malloc_alloc_122_23 }}	BOTTOM
	main	{{ (vector *)\$6_malloc_alloc_122_29 }}	ESCAPINGADDR	{{ (void *)\$6_malloc_alloc_122_29 }}	BOTTOM

# Concluzii: NO BUGS

Neterminare în versiuni anterioare ([Maroneze, 2017]):



# Neterminarea în versiuni anterioare

The screenshot displays the Monocypher IDE interface. The main editor shows a C program with several assertions. The first assertion is for a signed overflow: `h6 + c5 ≤ 9223372036854775808`. The second assertion is for a shift: `0 ≤ c5`. The third assertion is for a signed overflow: `h2 + (long long)((int)(1 << 25)) ≤ 92233720368`. The fourth assertion is for a shift: `(1 << 25) >> 26`. The program also includes a `h5 -= c5 << 25;` statement.

The right-hand pane shows the source code of `monocypher.c` with line numbers 865 to 876. The code defines variables `c0` through `c9` and `h0` through `h9` using bitwise operations.

The bottom pane shows the 'Callstack' tab. It contains a table with the following data:

Callstack	$0 \leq c5$	$0 \leq c5$ (after)	$c5$	$c5$ (after)
<code>ge_add ← ge_scalarmult ←</code>	invalid	unknown	$\in \{-1\}$	<BOTTOM>



Baudin, P. h. (2019).

FramaC.

site oficial.

online, accesat aprilie-mai 2019.



Cousot, P. (2008).

Abstract interpretation.

site oficial.

online, accesat aprilie-mai 2019.



FramaC (2019a).

ACSL.

site oficial.

online, accesat aprilie-mai 2019.



FramaC, G. (2019b).

Eva.

manual oficial.

online, accesat aprilie-mai 2019.



LoupVaillant (2017).

How I Implemented My Own Crypto.

eseu online.

online, accesat aprilie-mai 2019.



LoupVaillant (2019a).

Monocypher.

GitHub.

online, accesat aprilie-mai 2019.





LoupVaillant (2019b).

Monocypher.

site oficial.

online, accesat aprilie-mai 2019.



Maroneze, A. (2017).

A simple Eva tutorial.

blogul Framac.

online, accesat aprilie-mai 2019.