

DE LA RECHERCHE À L'INDUSTRIE

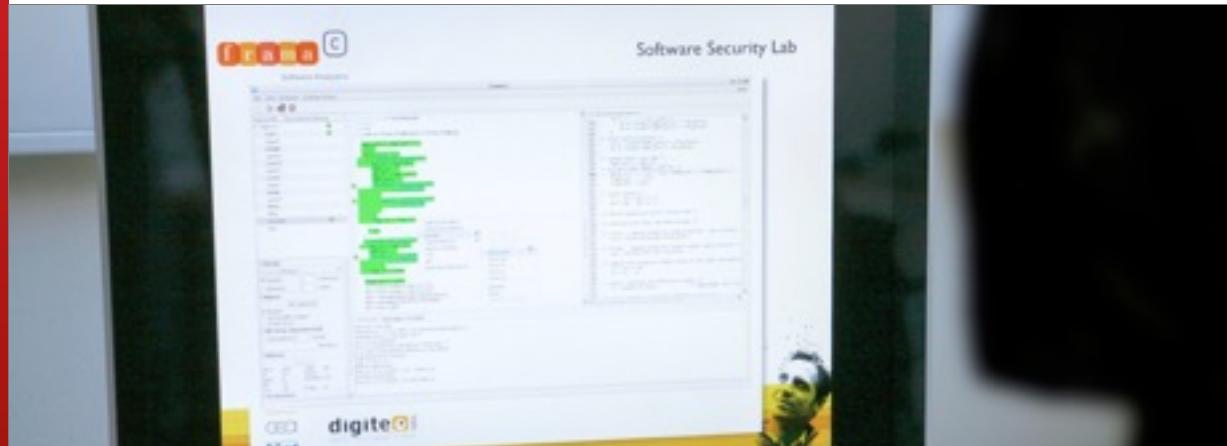


list

www.cea.fr

Proving a C-Code with GATeL

L. Correnson

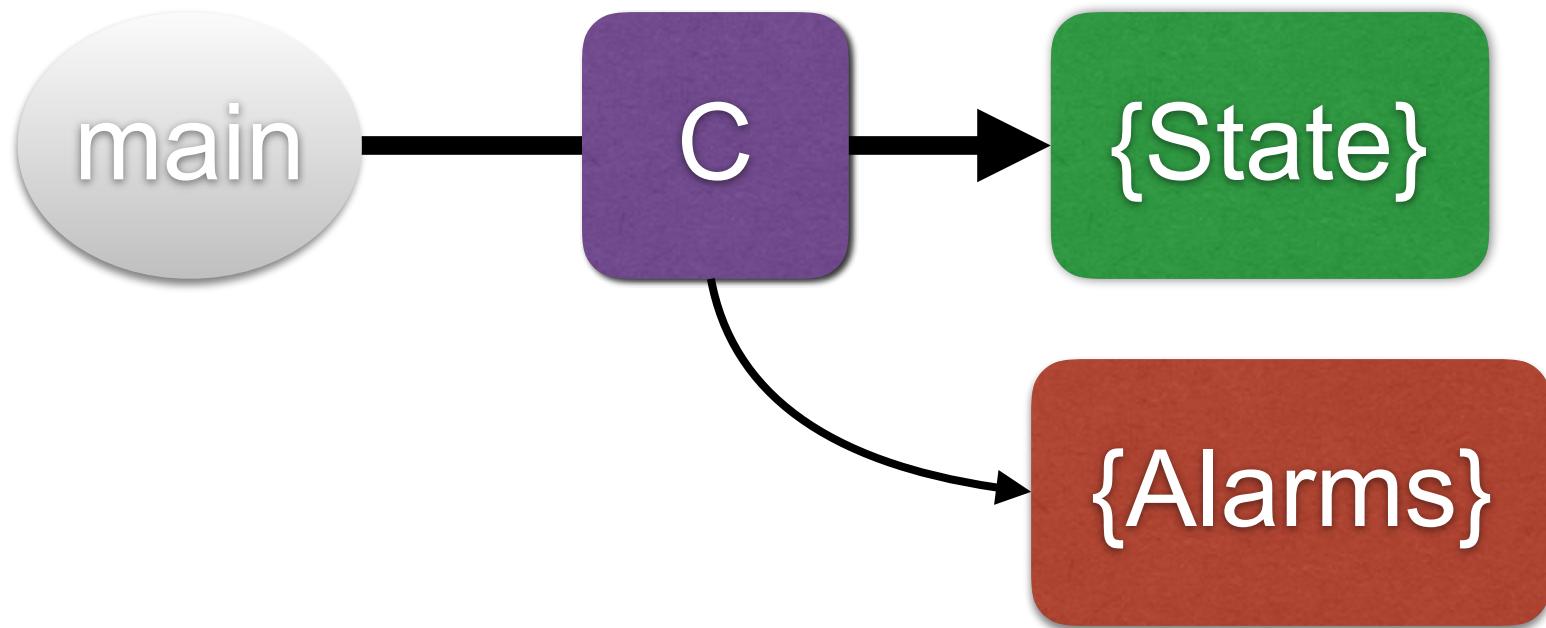


FRAMA-C DAY

WHAT'S THIS ABOUT ?

Value

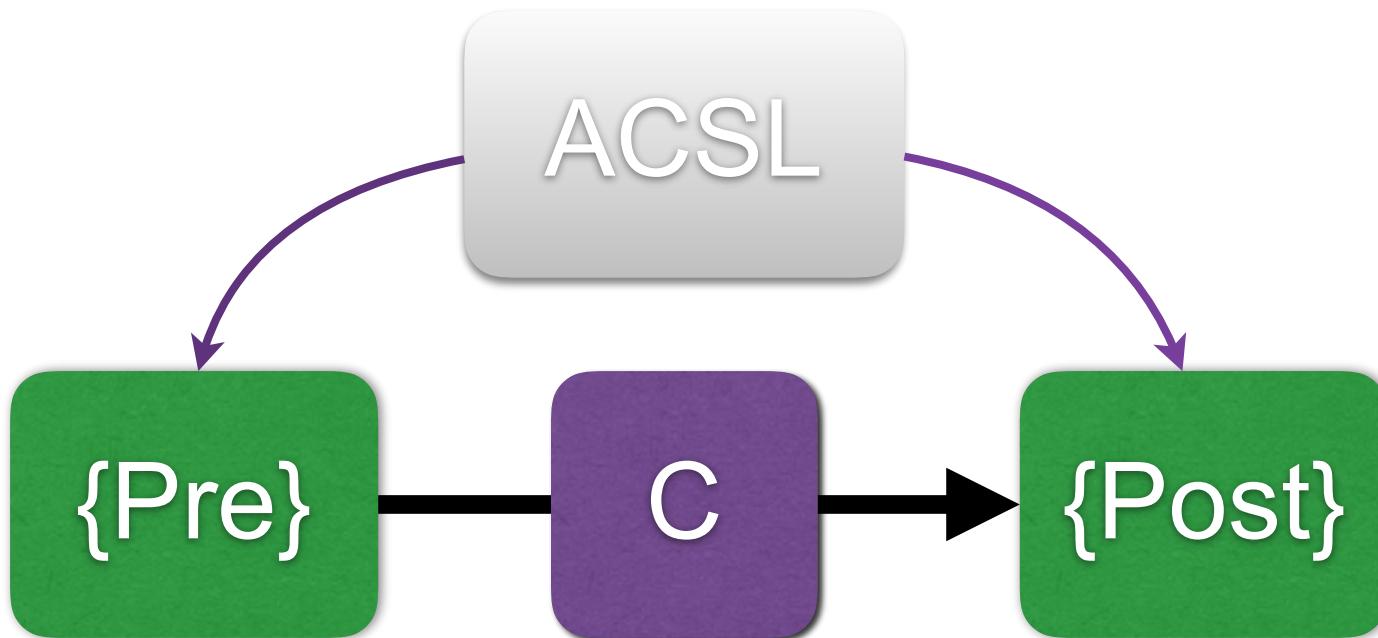
WHAT'S THIS ABOUT ?



WHAT'S THIS ABOUT ?

WP

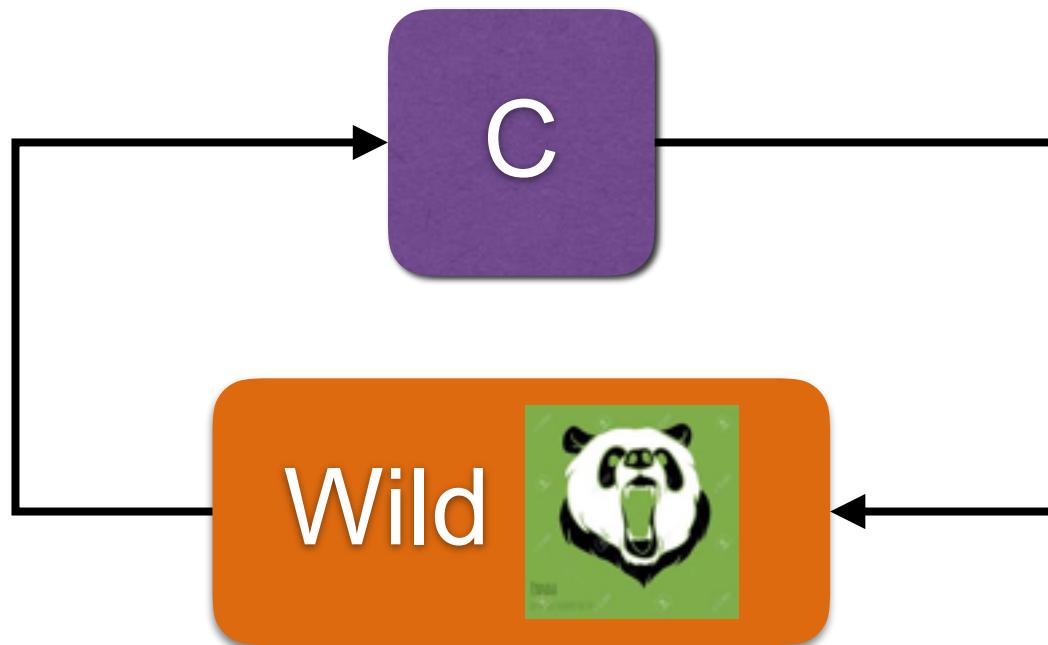
WHAT'S THIS ABOUT ?



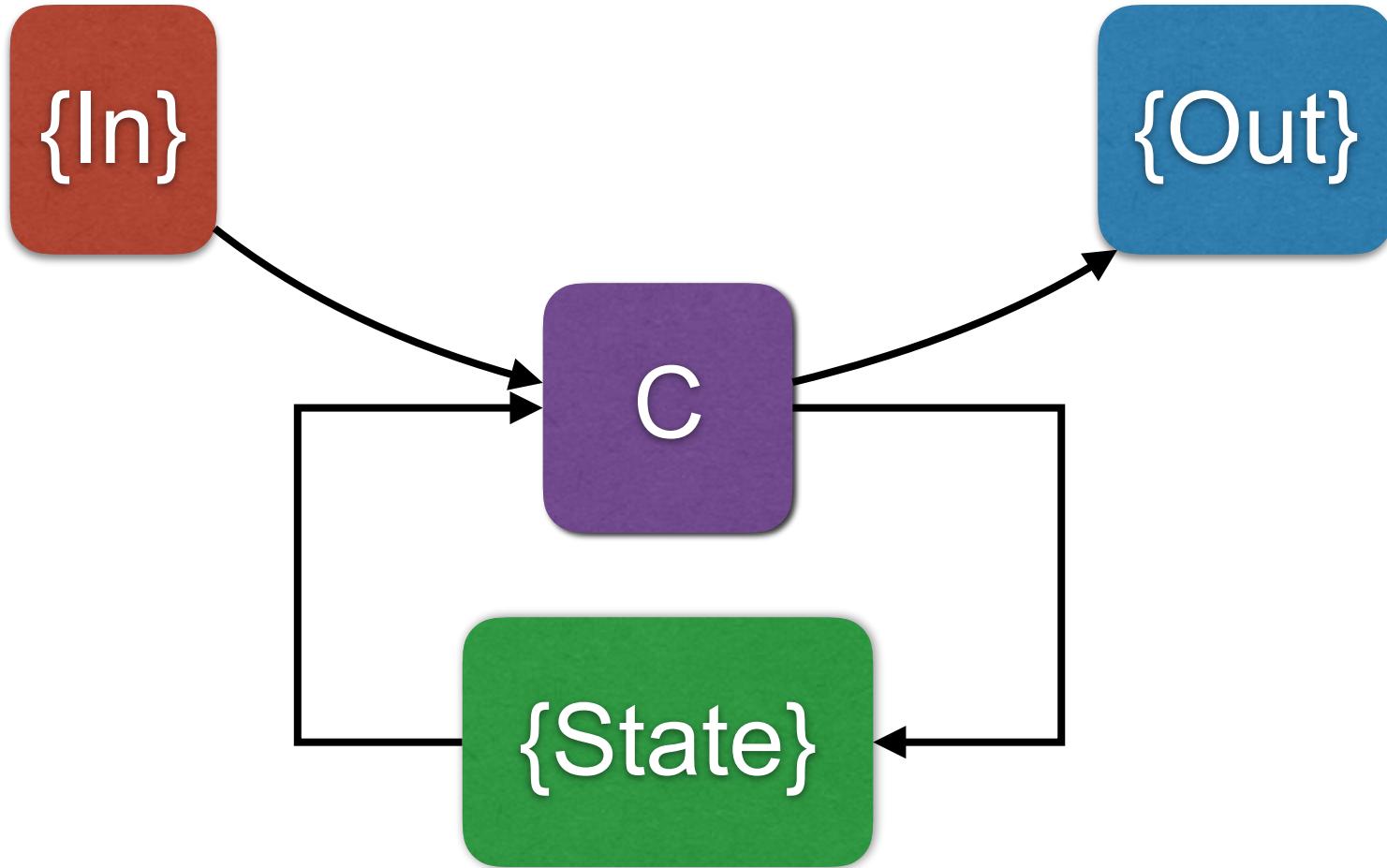
WHAT'S THIS ABOUT ?

Now ?

WHAT'S THIS ABOUT ?



CONTROL COMMAND LOOP



```
tel;  
let  
    never_since = if obs  
        then new  
        else (true  
tel;  
  
node HCS1(start,abort:bool)  
returns(end_tempo:bool;  
let  
    assert not(start and al  
    assert (true -> tempo :  
    assert tempo > tps; cyc  
  
(end_tempo,counter) = I  
(*! reach end_tempo  
and not(never  
and never_si  
tel;
```

Cycles

Constraints

Failures

Code

Lustre / Scade

Observer

Proof

Lustre / Scade

Unfolded

- At cycle

- At cycle 0:

split never{node 3}

. end tempo < true

Test Tree

Node 'Test'
'split', node 3, cycle 1'split', split_2
'if', node 1, cycle 0'->': initial
Test case 1'->': not_initial
Test case 2'if': true
'if', node 3, cycle 1'if': false
Test case 5'if': true
Test case 3'if': false
Test case 4

Close

CONTROL COMMAND LOOP

```
int main(void)
{
    init();
    while (1) {
        input();
        Compute();
        output();
    }
    return 0;
}
```

The diagram illustrates the execution flow of a control command loop. It shows a sequence of operations within a loop: `input()`, `Compute()`, and `output()`. Three arrows point from the text "Read globals from the Wild" to the `input()` call, indicating that global data is read into the loop. An arrow points from the `Compute()` call to the text "Update globals", indicating that the loop performs updates to global data. Another arrow points from the `output()` call to the text "Write globals to the Wild", indicating that global data is written out by the loop.

Read globals from the Wild

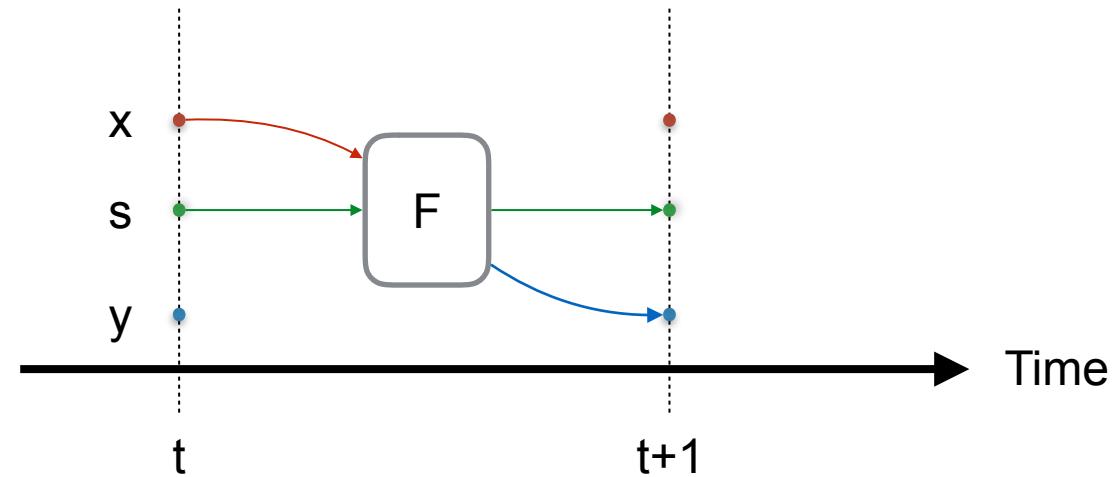
Update globals

Write globals to the Wild

THE « SYNCHRONE » PLUG-IN

```
init();
```

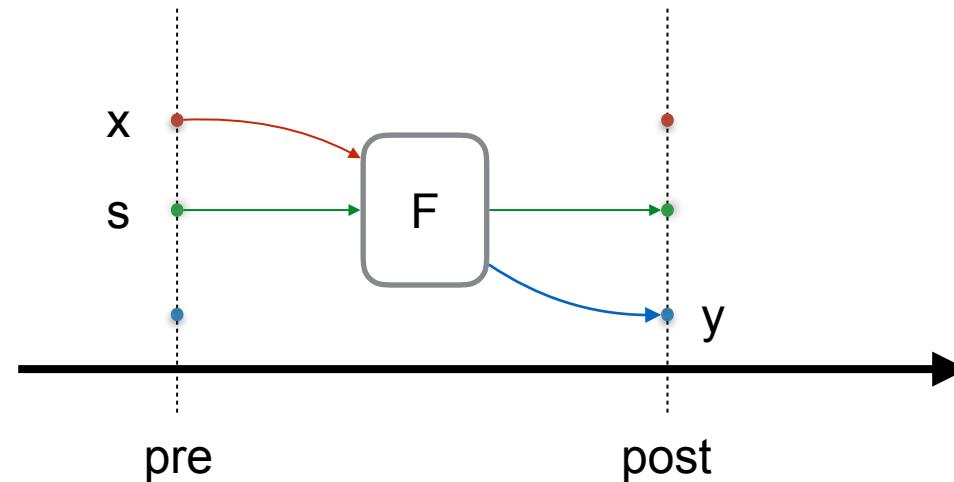
```
/*@ input x,... ;  
    probe y,... ;  
*/  
Compute();
```



A KIND OF WP ?

```
init();
```

```
/*@ ensures s = Fs(x, \old(s)) ;
   ensures y = Fy(x, \old(s)) ;
*/
Compute();
```



THE CLAIM

Just combine
WP & Value APIs

THE PROOF

```
[ correnson@~/Frama-C/dsync-core/src/plugins/sync ]  
$ wc -l *.ml  
    291 App.ml  
    508 Compiler.ml  
    150 Expr.ml  
     90 Extract.ml  
   198 Flow.ml  
   110 Fsync.ml  
    21 Gsync.ml  
   287 Lustre.ml  
     49 Msync.ml  
   146 Probe.ml  
1850 total
```

DEMO

Frama-C

File Project Analyses Help

Exit Source files Reparse Load session Save session Back Forward Analyses Stop

Source file

```

1 // Input: Teepartest(a);
2 probe Teepartest();
3 // Input: Teepartest(b); */
4 void cycle(void)
5 {
6     cap(A);
7     cap(B);
8     add(A);
9     add(B);
10    cap(C);
11    return;
12}
13
14 void add(blockt block *a) {
15     a->AC = a->AB;
16 }
17
18 void cap(blockt block *a) {
19     if (a->AB < -a->AC) a->AB = -a->AC;
20     else if (a->AB > a->AC) a->AB = a->AC;
21 }
22
23 struct block CAP_A = { AC : AB : NULL };
24 struct block CAP_B = { AC : AB : NULL };
25 struct block ADD_A = { AC : AB : NULL };
26 struct block CAP_C = { AC : AB : NULL };
27
28 void init(void)
29 {
30     p = 55;
31 }
32
33 // Input: a ; probe A; Input: b ;
34 void cycle(void)
35 {
36     cap(&cap_A);
37     cap(&cap_B);
38     add(&add_A);
39     add(&add_B);
40     cap(&cap_C);
41 }
42

```

+ WP

Model... Typed
(None)

Provers... All-edges (native)

RTE Split Trace

Invariants

Steps 0 Depth 0 Timeout 10 Process 4

Slicing Occurrence Metrics Impact Value

Run

0 stevel main main

Information Messages (1) Console Properties Values WF Goals

Multiple selections Expand rows Consolidated value Per callstack

Selection Callstack Add_A

main	.a IN {{&a}}
	.b IN {{&b}}
	.c IN {{&c}}

Qed

First Order Logic + Theories

Lang

Qed Instance with C/ACSL Definitions

Model

Vector of Logic Variables describing the C-Memory

CodeSemantics(Model)

Compiler of C-Code

LogicSemantics(Model)

Compiler of ACSL

Conditions

Formula Simplifier

```

open Cil_types
open Ctypes
open Lang.F

module Make(M : Memory.Model) :
sig

  open M

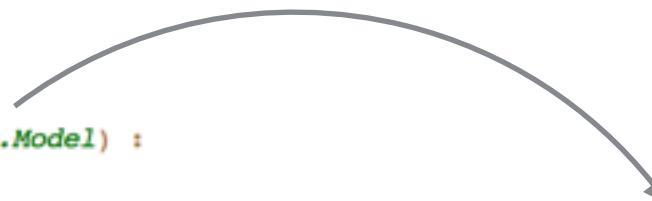
  type loc = M.loc
  type value = loc Memory.value
  type sigma = Sigma.t

  val cval : value -> term
  val cloc : value -> loc

  val cast : typ -> typ -> value -> value
  val equal_typ : typ -> value -> value -> pred
  val equal_obj : c_object -> value -> value -> pred

  val exp : sigma -> exp -> value
  val cond : sigma -> exp -> pred
  val lval : sigma -> lval -> loc

```



M.Sigma



M.loc

Cil.lval

M.value

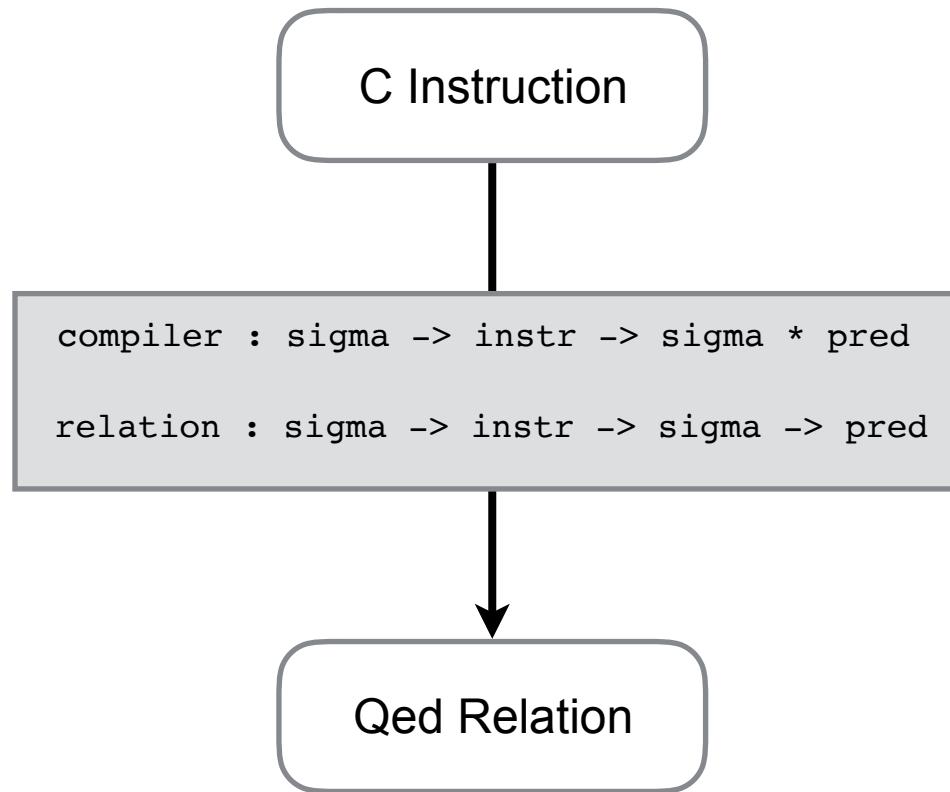
Cil.exp

```
let cc_store env obj loc value =
  let post = Sigma.havoc env.here (M.domain obj loc) in
  let seq = { pre = env.here ; post } in
  env.here <- post ; M.stored seq obj loc value

let cc_update env obj loc (e:exp) =
  match e.enode with
  | Lval p -> cc_copy env obj loc (CC.lval env.here p)
  | _ -> cc_store env obj loc (CC.val_of_exp env.here e)

let cc_assign env lv (e:exp) =
  let loc = CC.lval env.here lv in
  let obj = T.object_of (Cil.typeOfLval lv) in
  assume ~stmt:env.stmt (cc_update env obj loc e)
```

WP API : A SYMBOLIC EVALUATION ENGINE



Integer

Domain for Set of Integers

Value

Domain for Values : float, integer, address (Base x Integer), ...

OffsetMaps

Domain for Memory (Base x Offset => Integer)

Call Stacks

Context of the Analyzer

Value State

Call Stack => Lvalue => Value

VALUE API

```
let state callstack stmt =
  try
    let stk = Db.Value.get_stmt_state_callstack ~after:false stmt in
    match stk with
    | None -> None
    | Some hmap -> Some (Value_types.Callstack.Hashtbl.find hmap callstack)
  with Not_found -> None

let eval state lval =
  let d = snd (!Db.Value.eval_lval
               ~with_alarms:CILE.warn_none_mode None state lval) in
  try
    let (base,ival) = Locations.Location_Bytes.find_lonely_key d in
    match base,ival with
    | Base.Var(x,_), _ -> Addr(x,Ival.project_int ival)
    | Base.Null , Ival.Set [| k |] -> Int k
    | Base.Null , Ival.Float f ->
        let fa,fb = Fval.min_and_max f in
        let a = Fval.F.to_float fa in
        let b = Fval.F.to_float fb in
        let w = b -. a in
        if w < epsilon then Float (a +. 0.5 *. w) else Top
    | _ -> Top
  with Not_found | Ival.Not_Singleton_Int | Ival.Nan_or_infinite -> Top
```



SYNCHRONE IMPLEMENTATION

Main Loop

Pack init, inputs and compute in a loop

Run Value

Invariants on I-values, per callstack

Compile

Symbolic Evaluation of each instruction

- inline function calls
- value's invariants on each callstack
- simplify the equations

Lustre Expression

Another Qed Instance

Probe Extraction

Extract Output from Input and State

Memory Projection

Extract State from State

A SIMPLIFIED EXAMPLE

```
int c=1, g=0, *p=&g;
```

```
//@ input x ; probe y ;
void Compute(void) {
    *p += c ;
    y = *p + c ;
    *p += x ;
}
```

Value

```
C = 1 ; P = &G
```

WP

```
M1 = M0 [ P -> M0[P]+C ] ;
Y = M1[ P ] ;
M2 = M1 [ P -> M1[P]+X ] ;
```

Simplification

```
Y = M0[ &G ]+1 ;
M2 = M0[ &G -> X+Y ] ;
```

Projection

```
S0 => M0[ &G ]
S2 => M2[ &G ]
```

Extraction

```
Y = S0 + 1 ;
S2 = S0 + X + 1 ;
```

Lustre

```
Y = S + 1 ;
S = 0 -> pre (S + X + 1) ;
```

DEMO

Frama-C

File Project Analyses Help

Exit Source files Reparse Load session Save session Back Forward Analyses Stop

Source file: cycle.c

```

1 // Input: Keypressed(a);  

2 probe Keypressed(a);  

3 Input Keypressed(b); /*  

4 void cycle(void)  

5 {  

6     cap(A);  

7     cap(B);  

8     add(A);  

9     add(B);  

10    cap(C);  

11    return;  

12 }
13
14 void add(blockt block *a) {
15     a->AC = a->AB + a->BC;
16 }
17
18 void cap(blockt block *a) {
19     if (a->AB < -a->BC) a->AB = -a->BC;
20     else if (a->AB > a->BC) a->AB = a->BC;
21 }
22
23 struct block Cap_A = { AB : 0, BC : 0, AC : 0 };
24 struct block Cap_B = { AB : 0, BC : 0, AC : 0 };
25 struct block Add_A = { AB : 0, BC : 0, AC : 0 };
26 struct block Cap_C = { AB : 0, BC : 0, AC : 0 };
27
28 void init(void)
29 {
30     p = 55;
31 }
32
33 // Input: a ; probe A; Input: b ;
34 void cycle(void)
35 {
36     cap(&Cap_A);
37     cap(&Cap_B);
38     add(&Add_A);
39     add(&Add_B);
40     cap(&Cap_C);
41 }
```

WP

Model... Typed
(None)

Provers... All-edges (native)

RTE Split Trace

Invariants

Steps 0

Depth 0

Timeout 10

Process 4

Slicing Occurrence Metrics Impact Value

Run

0 stevel main main

Information Messages (1) Console Properties Values WP Goals

Multiple selections Selection Callstack Add_A

Expand rows Consolidated value Per callstack

Callstack Add_A

main .a IN {{&a}}
.b IN {{&b}}
.c IN {{&c}}

CONCLUSION

(just) combine

Frama-C APIs & other Tools

WP

Value

GATeL

FUTURE WORK

Qed

Soon on GitHub & Opam

WP

API for Symbolic Evaluation (helpers)

Value

Everything works (yes, I know, ...)

Doc

Oh, yes, of course ...

Synchrone

Maturation of the prototype
(loops, arrays, bitwise operators, modular extraction, ...)